

A Survey on Homomorphic Encryption Schemes: Theory and Implementation

ABBAS ACAR, HIDAYET AKSU, and A. SELCUK ULUAGAC, Florida International University
MAURO CONTI, University of Padua

Legacy encryption systems depend on sharing a key (public or private) among the peers involved in exchanging an encrypted message. However, this approach poses privacy concerns. The users or service providers with the key have exclusive rights on the data. Especially with popular cloud services, the control over the privacy of the sensitive data is lost. Even when the keys are not shared, the encrypted material is shared with a third party that does not necessarily need to access the content. Moreover, untrusted servers, providers, and cloud operators can keep identifying elements of users long after users end the relationship with the services. Indeed, *Homomorphic Encryption* (HE), a special kind of encryption scheme, can address these concerns as it allows any third party to operate on the encrypted data without decrypting it in advance. Although this extremely useful feature of the HE scheme has been known for over 30 years, the first plausible and achievable *Fully Homomorphic Encryption* (FHE) scheme, which allows any computable function to perform on the encrypted data, was introduced by Craig Gentry in 2009. Even though this was a major achievement, different implementations so far demonstrated that FHE still needs to be improved significantly to be practical on every platform. Therefore, this survey focuses on HE and FHE schemes. First, we present the basics of HE and the details of the well-known Partially Homomorphic Encryption (PHE) and Somewhat Homomorphic Encryption (SWHE), which are important pillars of achieving FHE. Then, the main FHE families, which have become the base for the other follow-up FHE schemes are presented. Furthermore, the implementations and new improvements in Gentry-type FHE schemes are also surveyed. Finally, further research directions are discussed. We believe this survey can give a clear knowledge and foundation to researchers and practitioners interested in knowing, applying, as well as extending the state of the art HE, PHE, SWHE, and FHE systems.

Categories and Subject Descriptors: E.3 [Data]: Data Encryption; K.6.5 [Management of Computing and Information Systems]: Security and Protection; K.4.1 [Computers and Society]: Public Policy Issues

General Terms: Encryption, Security, Privacy

Additional Key Words and Phrases: Fully homomorphic encryption, FHE, FHE implementation, FHE survey, Homomorphic Encryption, Partially Homomorphic Encryption, Somewhat Homomorphic Encryption, PHE, SWHE

This paper is an early draft of the survey that is being submitted to ACM CSUR and has been uploaded to arXiv for feedback from stakeholders.

1. INTRODUCTION

In ancient Greeks, the term "ὁμός" (homos) was used in the meaning of "same" while "μορφή" (morphe) was used for "shape" [Scott et al. 1940]. Then, the term *homomorphism* is coined and used in different areas. In abstract algebra, homomorphism is defined as a map preserving all the algebraic structures between the domain and range of an algebraic set [Malik et al. 2007]. The map is simply a function, i.e., an operation, which takes the inputs from the set of domain and outputs an element in the range, (e.g., addition, multiplication). In the cryptography field, the homomorphism is used as an encryption type. The *Homomorphic Encryption* (HE) is a kind of encryption scheme which allows a third party (e.g., cloud, service provider) to perform certain computable functions on the encrypted data while preserving the features of the function and format of the encrypted data. Indeed, this homomorphic encryption corresponds to a mapping in the abstract algebra. As an example for an additively HE scheme, for

Author's addresses: A. Acar, H. Aksu, and A. S. Uluagac, Electrical and Computer Engineering, Florida International University, Miami, FL-33199; emails: aacar001,haksu,suluagac@fiu.edu; M. Conti, Department of Mathematics, University of Padua, Padua, Italy and email: conti@math.unipd.it.

sample messages m_1 and m_2 , one can obtain $E(m_1 + m_2)$ by using $E(m_1)$ and $E(m_2)$ without knowing m_1 and m_2 explicitly, where E denotes the encryption function.

Normally, encryption is a crucial mechanism to preserve the privacy of any sensitive information. However, the conventional encryption schemes can not work on the encrypted data without decrypting it first. In other words, the users have to sacrifice their privacy to make use of cloud services such as file storing, sharing and collaboration. Moreover, untrusted servers, providers, popular cloud operators can keep physically identifying elements of users long after users end the relationship with the services [McMillan 2013]. This is a major privacy concern for users. In fact, it would be perfect if there existed a scheme which would not restrict the operations to be computed on the encrypted data while it would be still encrypted. From a historical perspective in cryptology, the term *homomorphism* is used for the first time by Rivest, Adleman, and Dertouzos [Rivest et al. 1978a] in 1978 as a possible solution to the computing without decrypting problem. This given basis in [Rivest et al. 1978a] has led to numerous attempts by researchers around the world to design such a homomorphic scheme with a large set of operations. In this work, our primary motivation is to survey the HE schemes focusing on the most recent improvements in this field, including *partially*, *somewhat*, and *fully* HE schemes.

A simple motivational HE example for a sample cloud application is illustrated in Figure 1. In this scenario, the client, C , first encrypts her private data (Step 1), then sends the encrypted data to the cloud servers, S , (Step 2). When the client wants to perform a function, $f()$, over her own data, she sends the function to server (Step 3). The server performs a homomorphic operation over the encrypted data using the *Eval* function, i.e., computes $f()$ blindfolded (Step 4) and returns the encrypted result to the client (Step 5). Finally, the client recovers the data with her own secret key and obtains $f(m)$ (Step 6). As seen in this simple example, the homomorphic operation, *Eval*, at the server side does not require the private key of the client and allows various operations such as addition and multiplication on the encrypted client data.

An early attempt to compute functions/operations on encrypted data is Yao's *garbled circuit*¹ study [Yao 1982]. Yao proposed two party communication protocol as a solution to the Millionaires' problem, which compares the wealth of two rich people without revealing the exact amount to each other. However, in Yao's *garbled circuit* solution, ciphertext size grows at least linearly with the computation of every gate in the circuit. This yields a very poor efficiency in terms of computational overhead and too much complexity in its communication protocol. Until Gentry's breakthrough in [Gentry 2009], all the attempts [Rivest et al. 1978b; Goldwasser and Micali 1982; ElGamal 1985; Benaloh 1994; Naccache and Stern 1998; Okamoto and Uchiyama 1998; Pailier 1999; Damgård and Jurik 2001; Kawachi et al. 2007; Yao 1982; Boneh et al. 2005; Sander et al. 1999; Ishai and Paskin 2007] have allowed either one type of operation or limited number of operations on the encrypted data. Moreover, some of the attempts are even limited over a specific type of set (e.g., branching programs). In fact, all these different HE attempts can neatly be categorized under three types of schemes with respect to the number of allowed operations on the encrypted data as follows: (1) *Partially Homomorphic Encryption* (PHE) allows only one type of operation with an unlimited number of times (i.e., no bound on the number of usages). (2) *Somewhat Homomorphic Encryption* (SWHE) allows some types of operations with a limited number of times. (3) *Fully Homomorphic Encryption* (FHE) allows unlimited number of operations with unlimited number of times.

¹A circuit is the set of connected gates (e.g., AND and XOR gates in boolean circuits), where the evaluation is completed by calculating the output of each gate in turn.

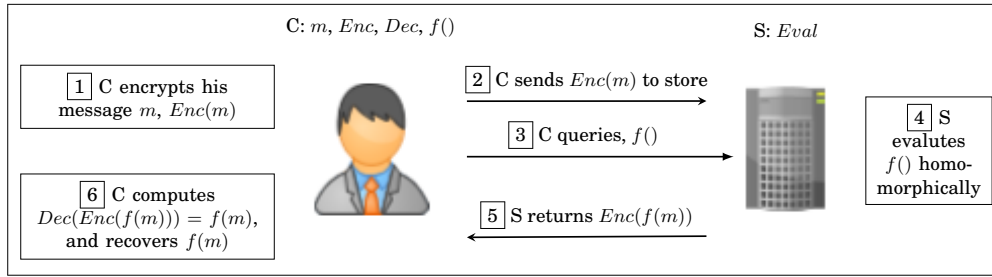


Fig. 1: A simple client-server HE scenario, where C is Client and S is Server

PHE schemes are deployed in some applications like e-voting [Benaloh 1987] or Private Information Retrieval (PIR) [Kushilevitz and Ostrovsky 1997]. However, these applications were restricted in terms of the types of homomorphic evaluation operations. In other words, PHE schemes can only be used for particular applications, whose algorithms include only addition or multiplication operation. On the other hand, the SWHE schemes support both addition and multiplication. Nonetheless, in SWHE schemes that are proposed before the first FHE scheme, the size of the ciphertexts grows with each homomorphic operation and hence the maximum number of allowed homomorphic operations is limited. These issues put a limit on the use of PHE and SWHE schemes in real-life applications. Eventually, the increasing popularity of cloud-based services accelerated the design of HE schemes which can support arbitrary number of homomorphic operations with random functions, i.e. FHE. Gentry's FHE scheme is the first plausible and achievable FHE scheme [Gentry 2009]. It is based on ideal-lattices in math and it is not only a description of the scheme, but also a powerful framework for achieving FHE. However, it is conceptually and practically not a realistic scheme. Especially, the *bootstrapping* part, which is the intermediate refreshing procedure of a processed ciphertext, is too costly in terms of computation. Therefore, a lot of follow-up improvements and new schemes were proposed in the following years.

Contribution: In our work, we basically aim to provide a comprehensive survey of all the main FHE schemes as of this writing. We also provide a survey of important PHE and SWHE schemes as they are the first works in accomplishing the FHE dream and are still popular as FHE schemes are very costly. Furthermore, we also survey the FHE implementations focusing on the improvements with each scheme. FHE attracts the interest of people from very different research areas in terms of theoretical, implementation, and application perspectives. We think that this survey provides an easy digest of the relatively complicated homomorphic encryption topic. For instance, while a mathematician focuses on the improvement in theoretical perspective, a hardware designer tries to improve the efficiency of FHE by implementing on GPU instead of CPU. All such different attempts make it harder to follow recent works. Because of this, we think that it is important to collect and categorize the existing FHE works focusing on recent improvements. In addition, this survey provides the challenges and future perspectives of HE to motivate the researchers and practitioners to explore and improve the performance of HE schemes and their applications. We believe this survey can give a clear knowledge foundation to researchers and practitioners interested in knowing, applying, as well as extending state of the art HE systems.

Organization: The remainder of the paper is organized as follows: In Section 2, descriptions of different HE schemes, PHE, SWHE, and FHE schemes are presented. Then, in Section 3, different implementations of SWHE and FHE schemes, which were introduced after Gentry's work, are given and their performances are discussed. Fi-

nally, in Section 4, further research directions and lessons learned are given and the paper is concluded.

2. RELATED WORK

Like our work in this paper, there are similar useful surveys in the literature. In fact, unfortunately some of the surveys only cover the theoretical information of the schemes as in [Parmar et al. 2014; Ahila and Shunmuganathan 2014] and some of them are directly for expert readers and mathematicians as in [Vaikuntanathan 2011; Silverberg 2013; Gentry 2014]. Compared to these surveys, our survey has a broad reader perspective including researchers and practitioners interested in the advances, implementations and applications in the field of HE, especially FHE. Furthermore, while the survey in [Aguilar-Melchor et al. 2013] only covers the signal processing applications, other in [Hrestak and Picek 2014] covers a few FHEs on only cloud applications. Since our survey is not limited to a specific application areas, we do not articulate these specific application areas in detail but we list the theory and implementation of all existing HE schemes, which can be used in possible futuristic application areas with recent advancements. After [Fontaine and Galand 2007] and [Akinwande 2009], many HE schemes were introduced. Compared to these useful surveys, our survey aims to focus on the most recent HE schemes, since most of the significant improvements are introduced recently (after 2009). Although [Moore et al. 2014b] is one of the most recent survey, it focuses on the hardware implementation solutions of FHE schemes. Our survey is not limited to hardware solutions, as, in addition to hardware solutions, it covers software solutions of implementations as well in the implementation section. Lastly, after [Sen 2013; Wu 2015], several new FHE schemes, which improves FHE in a sufficiently great way as to be worthy of attention, were proposed in the literature.

3. HOMOMORPHIC ENCRYPTION SCHEMES

In this section, we explain the basics of HE theory. Then, we present notable PHE, SWHE and FHE schemes. For each scheme, we also give a brief description of the scheme.

DEFINITION 1. *An encryption scheme is called homomorphic over an operation ' \star ' if it supports the following equation:*

$$E(m_1) \star E(m_2) = E(m_1 \star m_2), \quad \forall m_1, m_2 \in M, \quad (1)$$

where E is the encryption algorithm and M is the set of all possible messages.

In order to create an encryption scheme allowing the homomorphic evaluation of arbitrary function, it is sufficient to allow only addition and multiplication operations because addition and multiplication are functionally complete sets over finite sets. Particularly, any boolean circuit can be represented using only XOR (addition) and AND (multiplication) gates. While an HE scheme can use the same key for both encryption and decryption (symmetric), it can also be designed to use the different keys to encrypt and decrypt (asymmetric). A generic method to transform symmetric and asymmetric HE schemes to each other is demonstrated in [Rothblum 2011].

An HE scheme is primarily characterized by four operations: *KeyGen*, *Enc*, *Dec*, and *Eval*. *KeyGen* is the operation, which generates a secret and public key pair for asymmetric version of HE or a single key for symmetric version. Actually, *KeyGen*, *Enc* and *Dec* are not different from their classical tasks in conventional encryption schemes. However, *Eval* is an HE-specific operation, which takes ciphertexts as input and outputs a ciphertext corresponding to a functioned plaintext. *Eval* performs the function $f(\dots)$ over the ciphertexts (c_1, c_2) without seeing the messages (m_1, m_2) . *Eval* takes ciphertexts as input and outputs evaluated ciphertexts. The most crucial point

in this homomorphic encryption is that the format of the ciphertexts after an evaluation process must be preserved in order to be decrypted correctly. In addition, the size of the ciphertext should also be constant to support unlimited number of operations. Otherwise, the increase in the ciphertext size will require more resources and this will limit the number of operations.

Moreover, HE is computationally heavy and very complicated. There are different HE schemes in the literature to overcome this. Of all schemes in the literature, PHE schemes support *Eval* function for only either addition or multiplication, SWHE schemes support for only limited number of operations or some limited circuits (e.g., branching programs) while FHE schemes supports the evaluation of arbitrary functions (e.g., searching, sorting, max, min, etc.) with unlimited number of times over ciphertexts. The well-known PHE, SWHE, and FHE schemes are summarized in the timeline in Figure 2 and are explained in the following sections with a greater detail.

3.1. Partially Homomorphic Encryption Schemes

In this section, we articulate the details of PHE schemes. There are several useful PHE examples [Rivest et al. 1978b; Goldwasser and Micali 1982; ElGamal 1985; Benaloh 1994; Naccache and Stern 1998; Okamoto and Uchiyama 1998; Paillier 1999; Damgård and Jurik 2001; Kawachi et al. 2007] in the literature. Each has improved the PHE in some way. However, in this section, we primarily focus on major PHE schemes that are the basis for many other PHE schemes.

3.1.1. RSA. RSA is an early example of PHE and introduced by Rivest, Shamir, and Adleman [Rivest et al. 1978b] shortly after the invention of public key cryptography by Diffie Helman [Diffie and Hellman 1976]. RSA is the first feasible achievement of public key cryptosystem. Moreover, the homomorphic property of RSA was shown by Rivest, Adleman and Dertouzos [Rivest et al. 1978a] just after the seminal work of RSA. Indeed, the first attested use of the term "privacy homomorphism" is introduced in [Rivest et al. 1978a]. The security of the RSA cryptosystem is based on the hardness of *factoring problem* of the product of two large prime numbers [Gjøsteen 2004]. RSA is defined as follows:

- *KeyGen Algorithm*: First, for large primes p and q , $n = pq$ and $\phi = (p - 1)(q - 1)$ are computed. Then, e is chosen such that $\gcd(e, \phi)$ and d is calculated by computing the multiplicative inverse of e (i.e, $ed \equiv 1 \pmod{\phi}$). Finally, (e, n) is released as the public key pair while (d, n) is kept as the secret key pair.
- *Encryption Algorithm*: First, the message is converted into a plaintext m such that $0 \leq m < n$, then the RSA encryption algorithm is as follows:

$$c = E(m) = m^e \pmod{n}, \quad \forall m \in M, \quad (2)$$

where c is the ciphertext.

- *Decryption Algorithm*: The message m can be recovered from the ciphertext c using the secret key pair (d, n) as follows:

$$m = D(c) = c^d \pmod{n} \quad (3)$$

- *Homomorphic Property*: For $m_1, m_2 \in M$,

$$E(m_1) * E(m_2) = (m_1^e \pmod{n}) * (m_2^e \pmod{n}) = (m_1 * m_2)^e \pmod{n} = E(m_1 * m_2). \quad (4)$$

The homomorphic property of RSA shows that $E(m_1 * m_2)$ can be directly evaluated by using $E(m_1)$ and $E(m_2)$ without decrypting them. In other words, RSA is only homomorphic over multiplication. Hence, it does not allow the homomorphic addition of ciphertexts.

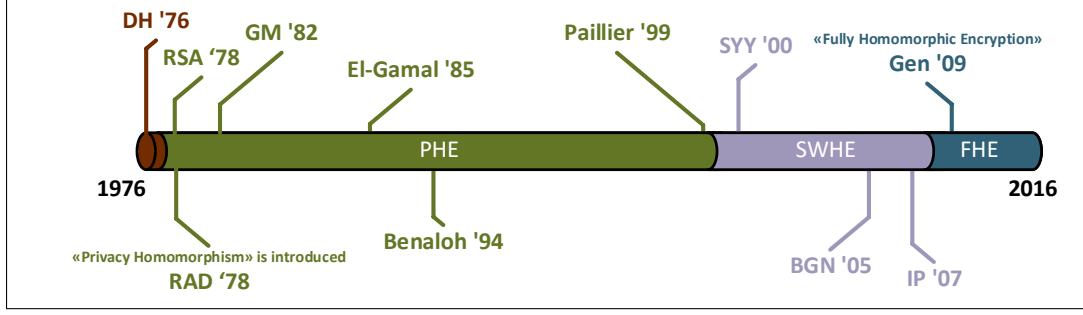


Fig. 2: Timeline of HE schemes until Gentry's first FHE scheme

3.1.2. Goldwasser-Micali. GM proposed the first probabilistic public key encryption scheme proposed in [Goldwasser and Micali 1982]. The GM cryptosystem is based on the hardness of *quadratic residuosity problem* [Gjøsteen 2004]. Number a is called *quadratic residue modulo n* if there exists an integer x such that $x^2 \equiv a \pmod{n}$. Quadratic residuosity problem decides whether a given number q is quadratic modulo n or not. GM cryptosystem is described as follows:

- **KeyGen Algorithm:** Similar to RSA, $n = pq$ is computed where p and q are distinct large primes and then, x is chosen as one of the quadratic nonresidue modulo n values with $\left(\frac{x}{n}\right) = 1$. Finally, (x, n) is published as the public key while (p, q) is kept as the secret key.
- **Encryption Algorithm:** Firstly, the message (m) is converted into a string of bits. Then, for every bit of the message m_i , a quadratic nonresidue value y_i is produced such that $\gcd(y_i, n) = 1$. Then, each bit is encrypted to c_i as follows:

$$c_i = E(m_i) = y_i^2 x^{m_i} \pmod{n}, \quad \forall m_i \in \{0, 1\}, \quad (5)$$

where $m = m_0 m_1 \dots m_r$, $c = c_0 c_1 \dots c_r$ and r is the block size used for the message space and x is picked from \mathbb{Z}_n^* at random for every encryption, where \mathbb{Z}_n^* is the multiplicative subgroup of integers modulo n which includes all the numbers smaller than n and relatively prime to n .

- **Decryption Algorithm:** Since x is picked from the set \mathbb{Z}_n^* ($1 < x \leq n-1$), x is quadratic residue modulo n for only $m_i = 0$. Hence, to decrypt the ciphertext c_i , one decides whether c_i is a quadratic residue modulo n or not; if so, m_i returns 0, else m_i returns 1.
- **Homomorphic Property:** For each bit $m_i \in \{0, 1\}$,

$$\begin{aligned} E(m_1) * E(m_2) &= (y_1^2 x^{m_1} \pmod{n}) * (y_2^2 x^{m_2} \pmod{n}) \\ &= (y_1 * y_2)^2 x^{m_1 + m_2} \pmod{n} = E(m_1 + m_2). \end{aligned} \quad (6)$$

The homomorphic property of the GM cryptosystem shows that encryption of the sum $E(m_1 \oplus m_2)$ can be directly calculated from the separately encrypted bits, $E(m_1)$ and $E(m_2)$. Since the message and ciphertext are the elements of the set $\{0, 1\}$, the operation is the same with exclusive-OR (XOR)². Hence, GM is homomorphic over only addition for binary numbers.

3.1.3. Elgamal. In 1985, Taher Elgamal proposed a new public key encryption scheme [ElGamal 1985] which is the improved version of the original Diffie-Hellman

²XOR can be thought as binary addition.

Key Exchange [Diffie and Hellman 1976] algorithm, which is based on the hardness of certain problems in discrete logarithm [Gjøsteen 2004]. It is mostly used in hybrid encryption systems to encrypt the secret key of a symmetric encryption system. The Elgamal cryptosystem is defined as follows:

- *KeyGen Algorithm*: A cyclic group G with order n using generator g is produced. In a cyclic group, it is possible to generate all the elements of the group using the powers of one of its own element. Then, $h = g^y$ computed for randomly chosen $y \in \mathbb{Z}_n^*$. Finally, the public key is (G, n, g, h) and x is the secret key of the scheme.
- *Encryption Algorithm*: The message m is encrypted using g and x , where x is randomly chosen from the set $\{1, 2, \dots, n-1\}$ and the output of the encryption algorithm is a ciphertext pair $(c = (c_1, c_2))$:

$$c = E(m) = (g^x, mh^x) = (g^x, mg^{xy}) = (c_1, c_2), \quad (7)$$

- *Decryption Algorithm*: To decrypt the ciphertext c , first, $s = c_1^y$ is computed where y is the secret key. Then, decryption algorithm works as follows:

$$c_2 \cdot s^{-1} = mg^{xy} \cdot g^{-xy} = m. \quad (8)$$

- *Homomorphic Property*:

$$E(m_1) * E(m_2) = (g^{x_1}, m_1 h^{x_1}) * (g^{x_2}, m_2 h^{x_2}) = (g^{x_1+x_2}, m_1 * m_2 h^{x_1+x_2}) = E(m_1 * m_2). \quad (9)$$

As seen from this derivation, the Elgamal cryptosystem is multiplicatively homomorphic. It does not support addition operation over ciphertexts.

3.1.4. Benaloh. Benaloh proposed an extension of the GM Cryptosystem by improving it to encrypt the message as a block instead of bit by bit [Benaloh 1994]. Benaloh's proposal was based on the higher residuosity problem. Higher residuosity problem (x^n) [Gjøsteen 2004] is the generalization of quadratic residuosity problems (x^2) that is used for the GM cryptosystem.

- *KeyGen Algorithm*: Block size r and large primes p and q are chosen such that r divides $p-1$ and r is relatively prime to $(p-1)/r$ and $q-1$ (i.e., $\gcd(r, (p-1)/r) = 1$ and $\gcd(r, (q-1)) = 1$). Then, $n = pq$ and $\phi = (p-1)(q-1)$ are computed. Lastly, $y \in \mathbb{Z}_n^*$ is chosen such that $y^\phi \not\equiv 1 \pmod{n}$, where \mathbb{Z}_n^* is the multiplicative subgroup of integers modulo n which includes all the numbers smaller than n and relatively prime to n . Finally, (y, n) is published as public key, and (p, q) is kept as secret key.
- *Encryption Algorithm*: For the message $m \in \mathbb{Z}_r$, where $\mathbb{Z}_r = \{0, 1, \dots, r-1\}$, choose a random u such that $u \in \mathbb{Z}_n^*$. Then, to encrypt the message m :

$$c = E(m) = y^m u^r \pmod{n}, \quad (10)$$

where the public key is the modulus n and base y with the block size of r .

- *Decryption Algorithm*: The message m is recovered by an exhaustive search for $i \in \mathbb{Z}_r$ such that

$$(y^{-i} c)^{\phi/r} \equiv 1, \quad (11)$$

where the message m is returned as the value of i , i.e., $m = i$.

- *Homomorphic Property*:

$$\begin{aligned} E(m_1) * E(m_2) &= (y^{m_1} u_1^r \pmod{n}) * (y^{m_2} u_2^r \pmod{n}) \\ &= y^{m_1+m_2} (u_1 * u_2)^r \pmod{n} = E(m_1 + m_2 \pmod{n}). \end{aligned} \quad (12)$$

Homomorphic property of Benaloh shows that any multiplication operation on encrypted data corresponds to the addition on plaintext. As the encryption of the addition

of the messages can directly be calculated from encrypted messages $E(m_1)$ and $E(m_2)$, the Benaloh cryptosystem is additively homomorphic.

3.1.5. Paillier. In 1999, Paillier [Paillier 1999] introduced another novel probabilistic encryption scheme based on *composite residuosity problem* [Gjøsteen 2004]. *Composite residuosity problem* is very similar to quadratic and higher residuosity problems that are used in GM and Benaloh cryptosystems. It questions whether there exists an integer x such that $x^n \equiv a \pmod{n^2}$ for a given integer a .

- **KeyGen Algorithm:** For large primes p and q such that $\gcd(pq, (p-1)(q-1)) = 1$, compute $n = pq$ and $\lambda = \text{lcm}(p-1, q-1)$. Then, select a random integer $g \in \mathbb{Z}_{n^2}^*$ by checking whether $\gcd(n, L(g^\lambda \pmod{n^2})) = 1$, where the function L is defined as $L(u) = (u-1)/n$ for every u from the subgroup $Z_{n^2}^*$ which is multiplicative subgroup of integers modulo n^2 instead of n like in the Benaloh cryptosystem. Finally, the public key is (n, g) and the secret key is (p, q) pair.
- **Encryption Algorithm:**
For each message m , the number r is randomly chosen and the encryption works as follows:

$$c = E(m) = g^m r^n \pmod{n^2}, \quad (13)$$

- **Decryption Algorithm:** For a proper ciphertext $c < n^2$, the decryption is done by:

$$D(c) = \frac{L(c^\lambda \pmod{n^2})}{L(g^\lambda \pmod{n^2})} \pmod{n} = m, \quad (14)$$

where private key pair is (p, q) .

- **Homomorphic Property:**

$$\begin{aligned} E(m_1) * E(m_2) &= (g^{m_1} r_1^n \pmod{n^2}) * (g^{m_2} r_2^n \pmod{n^2}) \\ &= g^{m_1+m_2} (r_1 * r_2)^n \pmod{n^2} = E(m_1 + m_2). \end{aligned} \quad (15)$$

This derivation shows that Paillier's encryption scheme is homomorphic over addition. In addition to homomorphism over the addition operation, Paillier's encryption scheme has some additional homomorphic properties, which allow extra basic operations on plaintexts $m_1, m_2 \in \mathbb{Z}_{n^2}^*$ by using the encrypted plaintexts $E(m_1), E(m_2)$ and public key pair (n, g) :

$$E(m_1) * E(m_2) \pmod{n^2} = E(m_1 + m_2 \pmod{n}), \quad (16)$$

$$E(m_1) * g^{m_2} \pmod{n^2} = E(m_1 + m_2 \pmod{n}), \quad (17)$$

$$E(m)^k \pmod{n^2} = E(km \pmod{n}). \quad (18)$$

These additional homomorphic properties (e.g., scalar multiplication in Equation (18)) describe different cross-relation between different operations on the encrypted data and the plaintexts. In other words, Equations (16), (17), and (18) show how the operation computed on encrypted data affects the plaintexts.

3.1.6. Others. Moreover, Okamoto-Uchiyama (OU) [Okamoto and Uchiyama 1998] proposed a new PHE scheme to improve the computational performance by changing the set, where the encryptions of previous HE schemes work. The domain of the scheme is the same as the previous public key encryption schemes, \mathbb{Z}_n^* , however Okamoto-Uchiyama sets $n = p^2q$ for large primes p and q . Furthermore, Naccache-Stern (NS) [Naccache and Stern 1998] presented another PHE scheme as a generalization of Benaloh cryptosystem to increase its computational efficiency. The proposed

Table I: Homomorphic properties of well-known PHE schemes

Scheme	Homomorphic Operation		
	Add	Mult	Extra
RSA [Rivest et al. 1978b]		✓	
GM [Goldwasser and Micali 1982]	✓		
Elgamal [ElGamal 1985]		✓	
Benaloh [Benaloh 1994]	✓		
NS [Naccache and Stern 1998]	✓		
OU [Okamoto and Uchiyama 1998]	✓		
Paillier [Paillier 1999]	✓		✓
DJ [Damgård and Jurik 2001]	✓		✓
KTX [Kawachi et al. 2007]	✓		
Galbraith [Galbraith 2002]	✓		✓

work changed only the decryption algorithm of the scheme. Likewise, Damgård-Jurik (DJ) [Damgård and Jurik 2001] introduced another PHE scheme as a generalization of Paillier. These three cryptosystems preserve the homomorphic property while improving the original homomorphic schemes.

Similarly, Kawachi (KTX) et al. [Kawachi et al. 2007] suggested an additively homomorphic encryption scheme over a large cyclic group, which is based on the hardness of underlying lattice problems. They named the homomorphic property of their proposed scheme as *pseudohomomorphic*. Pseudohomomorphism is an algebraic property and still allows homomorphic operations on ciphertext, however the decryption of the homomorphically operated ciphertext works with a small decryption error. Finally, Galbraith [Galbraith 2002] introduced a more natural generalization of Paillier’s cryptosystem applying it on elliptic curves while still preserving the homomorphic property of the Paillier’s cryptosystem. Homomorphic properties of well-known PHE schemes are briefly summarized in Table I.

3.2. Somewhat Homomorphic Encryption Schemes

In this section, we explain the SWHE schemes in detail. There are useful SWHE examples [Yao 1982; Sander et al. 1999; Boneh et al. 2005; Ishai and Paskin 2007] in the literature before 2009. After the first plausible FHE published in 2009 [Gentry 2009], some SWHE versions of FHE schemes were also proposed because of the performance issues associated with FHE schemes. We cover the FHE schemes under the FHE section. In this section, we primarily focus on major SWHE schemes, which were used as a stepping stone to the first plausible FHE scheme.

3.2.1. BGN. Before 2005, all proposed cryptosystems’ homomorphism properties were restricted to only either addition or multiplication operation i.e., SWHE schemes. One of the most significant steps toward an FHE scheme was introduced by Boneh-Goh-Nissim (BGN) in [Boneh et al. 2005]. BGN evaluates 2-DNF³ formulas on ciphertext and it supports arbitrary number of additions and one multiplication by keeping the ciphertext size constant. The hardness of the scheme is based on the *subgroup decision problem* [Gjøsteen 2004]. Subgroup decision problem simply decides whether an element is a member of a subgroup G_p of group G of composite order $n = pq$, where p and q are distinct primes.

³Disjunctive Normal Form with at most 2 literals in each clause.

- KeyGen Algorithm*: Public key is released as (n, G, G_1, e, g, h) . In the public key, e is a bilinear map such that $e : G \times G \rightarrow G_1$, where G, G_1 are groups of order $n = q_1 q_2$. g and u are the generators of G and set $h = u^{q_2}$ and h is the generator of G with order q_1 , which is kept hidden as the secret key.
- Encryption Algorithm*: To encrypt a message m , a random number r from the set $\{0, 1, \dots, n-1\}$ is picked and encrypted using the precomputed g and h as follows:

$$c = E(m) = g^m h^r \pmod n \quad (19)$$

- Decryption Algorithm*: To decrypt the ciphertext c , one firstly computes $c' = c^{q_1} = (g^m h^r)^{q_1} = (g^{q_1})^m$ (Note that $h^{q_1} \equiv 1 \pmod n$) and $g' = g^{q_1}$ using the secret key q_1 and decryption is completed as follows:

$$m = D(c) = \log_{g'} c' \quad (20)$$

In order to decrypt efficiently, the message space should be kept small because of the fact that discrete logarithm can not be computed quickly.

- Homomorphism over Addition*: Homomorphic addition of plaintexts m_1 and m_2 using ciphertexts $E(m_1) = c_1$ and $E(m_2) = c_2$ are performed as follows:

$$c = c_1 c_2 h^r = (g^{m_1} h^{r_1})(g^{m_2} h^{r_2}) h^r = g^{m_1+m_2} h^{r'}, \quad (21)$$

where $r = r_1 + r_2 + r$ and it can be seen that $m_1 + m_2$ can be easily recovered from the resulting ciphertext c .

- Homomorphism over Multiplication*: To perform homomorphic multiplication, use g_1 with order n and h_1 with order q_1 and set $g_1 = e(g, g)$, $h_1 = e(g, h)$, and $h = g^{\alpha q_2}$. Then, the homomorphic multiplication of messages m_1 and m_2 using the ciphertexts $c_1 = E(m_1)$ and $c_2 = E(m_2)$ are computed as follows:

$$\begin{aligned} c &= e(c_1, c_2) h_1^r = e(g^{m_1} h^{r_1}, g^{m_2} h^{r_2}) h_1^r \\ &= g_1^{m_1 m_2} h_1^{m_1 r_2 + r_2 m_1 + \alpha q_2 r_1 r_2 + r} = g_1^{m_1 m_2} h_1^{r'} \end{aligned} \quad (22)$$

It is seen that r' is uniformly distributed like r and so $m_1 m_2$ can be correctly recovered from resulting ciphertext c . However, c is now in the group G_1 instead of G . Therefore, another homomorphic multiplication operation is not allowed in G_1 because there is no pairing from the set G_1 . However, resulting ciphertext in G_1 still allows unlimited number of homomorphic additions. Moreover, Boneh et al. also showed the evaluation of 2-DNF formulas using the basic 2-DNF protocol. Their protocol gives a quadratic improvement in terms of the protocol complexity over Yao's well-known garbled circuit protocol in [Yao 1982].

3.2.2. Others. Another idea of evaluating operations on encrypted data is realized over different sets. Sander, Young, and Yung (SYU) described first SWHE scheme over a semi-group, NC^1 ,⁴ [Sander et al. 1999], which requires less properties than a group. NC^1 is a complexity class which includes the circuits with poly-logarithmic depth and polynomial size. The proposed scheme supported polynomially many ANDing of ciphertexts with one OR/NOT gate. However, the ciphertext size increased by a constant multiplication with each OR/NOT gate evaluation. This increase limits the evaluation of circuit depth. Yuval Ishai and Anat Paskin (IP) expanded the set to branching programs (aka Binary Decision Diagrams), which are the directed acyclic graphs where every nodes have two outgoing edges with labelled binary 0 and 1 [Ishai and Paskin 2007]. In other words, they proposed a public key encryption scheme by evaluating the branching programs on the encrypted data. Moreover, Melchor et al. [Melchor

⁴NC stands for "Nick's Class" for the honor of Nick Pippenger

Table II: Comparison of some well-known SWHE schemes before Gentry's work

	Evaluation Size	Evaluation Circuit	Ciphertext Size
Yao [Yao 1982]	arbitrary	garbled circuit	grows at least linearly
SYN [Sander et al. 1999]	polly-many AND & one OR/NOT	NC^1 circuit	grows exponentially
BGN [Boneh et al. 2005]	unlimited add & 1 mult	2-DNF formulas	constant
IP [Ishai and Paskin 2007]	arbitrary	branching programs	doesn't depend on the size of function

et al. 2010] proposed a generic construction method to obtain a chained encryption scheme allowing the homomorphic evaluation of constant depth circuit over ciphertext. The chained encryption scheme is obtained from well-known encryption schemes with some homomorphic properties. For example, they showed how to obtain a combination of BGN [Boneh et al. 2005] and Kawachi et al. [Kawachi et al. 2007]. As mentioned before, BGN allows arbitrary number of additions and one multiplication while Kawachi's scheme is only additively homomorphic. Hence, the resulting combined scheme allows arbitrary additions and two multiplications. They also showed how to apply this procedure on the scheme in [Melchor et al. 2008] allowing predefined number of homomorphic additions, to obtain a scheme which allows arbitrary number of multiplications as well. However, in multiplication, ciphertext size grows exponentially while it is constant in a homomorphic addition. The summary of some well-known SWHE schemes is given in Table II. As shown in Table II, while in Yao, SYN, and IP cryptosystems, the size of the ciphertext grows with each homomorphic operation, in BGN it stays constant. This property of BGN is a significant improvement to obtain an FHE scheme. Accordingly, Gentry, Halevi, and Vaikuntanathan later simplified the BGN cryptosystem [Gentry et al. 2010]. In their version, the underlying security assumption is changed to hardness of the LWE problem. The BGN cryptosystem chooses input from a small set to decrypt correctly. In contrast, a recent scheme introduced in [Gentry et al. 2010] have much larger message space. Moreover, some of the attempts to obtain an FHE scheme based on SWHE schemes are reported as broken. For instance, vulnerabilities for [Mullen and Shiue 1994; i Ferrer 1996; Grigoriev and Ponomarenko 2006; Domingo-Ferrer 2002] were reported in [Steinwandt and Geiselmann 2002; Choi et al. 2007; Wagner 2003; Cheon et al. 2006], respectively.

3.3. Fully Homomorphic Encryption Schemes

An encryption scheme is called Fully Homomorphic Encryption (FHE) scheme if it allows unlimited number of evaluation operations on the encrypted data and resulting output is within the ciphertext space. After almost 30 years from the introduction of privacy homomorphism concept [Rivest et al. 1978a], Gentry presented the first feasible proposal in his seminal PhD thesis to a long term open problem, which is obtaining an FHE scheme [Gentry 2009]. Gentry's proposed scheme gives not only an FHE scheme, but also a general framework to obtain an FHE scheme. Hence, a lot of researchers have attempted to design a secure and practical FHE scheme after Gentry's work.

Although Gentry's proposed ideal lattice-based FHE scheme [Gentry 2009] is very promising, it also had a lot of bottlenecks such as its computational cost in terms of applicability in real life and some of its advanced mathematical concepts make it diffi-

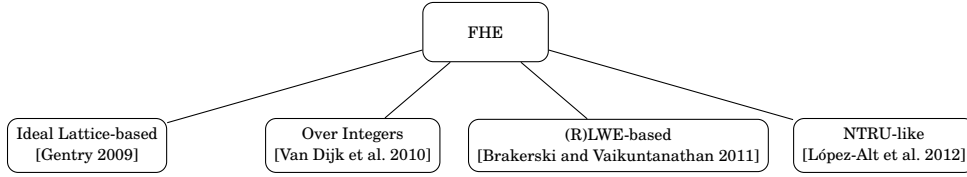


Fig. 3: Main FHE families after Gentry's breakthrough

cult for average users to understand. Therefore, many new schemes and optimizations have followed his work in order to address aforementioned bottlenecks. The security of new approaches to obtain a new FHE scheme is mostly based on the problems on lattices.

A lattice is the linear combinations of independent vectors (basis vectors), b_1, b_2, \dots, b_n . A lattice L is formulated as follows:

$$L = \sum_{i=1}^n \vec{b}_i * v_i, \quad v_i \in \mathbb{Z}, \quad (23)$$

where each vectors b_1, b_2, \dots, b_i is called a basis of the lattice L . The basis of a lattice is not unique. There are infinitely many bases for a given lattice. A basis is called "good" if the basis vectors are almost orthogonal and, otherwise it is called "bad" basis of the lattice. Roughly, while good bases are typically long, bad bases are relatively shorter. Indeed, the lattice theory is firstly presented by Minkowski [Minkowski 1968]. Then as a seminal work, Ajtai mentioned a class of random worst-case lattice problem in [Ajtai 1996]. Two well-known modern problems suggested in [Ajtai 1996] for lattice-based cryptosystems are Closest Vector Problem (CVP) and Shortest Vector Problem (SVP) [Peikert 2015]. A year after, Goldreich, Goldwasser, and Halevi (GGH) [Goldreich et al. 1997] proposed an important type of PKE scheme, whose hardness is based on the *lattice reduction problems* [Peikert 2015]. Lattice reduction tries to find a good basis, which is relatively short and orthogonal, for a given lattice. In GGH cryptosystem, the public key and the secret key is chosen from "bad" and "good" basis of the lattice, respectively. The idea behind this choice is that CVP and SVP problems can easily be solved in polynomial time for the lattices with the known good bases however best known algorithms (for example LLL in [Lenstra et al. 1982]) solve these problems in exponential time without knowing the good bases of the lattice. Hence, recovering the message from a given ciphertext is equal to solve the CVP and SVP problems in a practical time. Then, in GGH cryptosystem, the message is embedded to the noise. Finally, the noise is added to a lattice point to obtain the ciphertext. In order to recover the message from ciphertext, the private key is used to find the closest lattice point. Before Gentry's work, in [Regev 2006], cryptographers' attention is drawn to lattice-based cryptology and especially its great promising properties for post-quantum cryptology. Its promising properties are listed as its security proofs, efficient implementations, and simplicity. Moreover, another lattice-related problem, which gains popularity in last few years, especially after being used as a base to build a FHE scheme is LWE [Zhang 2014]. In addition, one of the most significant works for lattice-based cryptosystems was studied in [Hoffstein et al. 1998]. They presented a new PKE scheme, whose security is based on SVP on the lattice. In the SVP problem, given a basis of a lattice, the goal is to find the shortest nonzero vector in the lattice.

After Gentry's work, the lattices have become more popular among cryptography researchers. First, some works like [Smart and Vercauteren 2010] focused on just improving Gentry's ideal lattice-based FHE scheme in [Gentry 2009]. Then, an FHE

scheme over integers based on the Approximate-GCD problems is introduced [Van Dijk et al. 2010]. The main motivation behind the scheme is the conceptual simplicity. Afterwards, another FHE scheme whose hardness based on Ring Learning with Error (RLWE) problems is suggested [Brakerski and Vaikuntanathan 2011]. The proposed scheme promises some efficiency features. Lastly, an NTRU-like FHE is presented for its promising efficiency and standardization properties [López-Alt et al. 2012]. NTRU-Encrypt is an old and strongly standardized lattice-based encryption scheme whose homomorphic properties are realized recently. So, these and similar attempts can be categorized into under four main FHE families as shown in Figure 3: (1) Ideal lattice-based [Gentry 2009], (2) Over integers [Van Dijk et al. 2010], (3) (R)LWE-based [Brakerski and Vaikuntanathan 2011], and (4) NTRU-like [López-Alt et al. 2012]. In the following sections, we will articulate these four main FHE families in greater detail. And, we will also explore other follow-up works after these.

3.3.1. Ideal Lattice-based FHE schemes. Gentry's first FHE scheme in his PhD thesis [Gentry 2009] is a GGH-type of encryption scheme proposed originally by Goldreich et al. [Goldreich et al. 1997]. However, Gentry encrypted the message by embedding noise using double layer instead of one layer idea in GGH cryptosystem. Indeed, Gentry started his breakthrough work from SWHE scheme based on ideal lattices.

As mentioned earlier, an SWHE scheme can evaluate the ciphertext homomorphically for only a limited number of operations. After a certain threshold, the decryption function fails to recover the message from the ciphertext correctly. The amount of noise in the ciphertext must be decreased to transform the noisy ciphertext into a proper ciphertext. Gentry used genius blueprint methods called *squashing* and *bootstrapping* to obtain a ciphertext which allows a number of homomorphic operations to be performed on it. This processes can be repeated again and again. In other words, one can evaluate unlimited operations on the ciphertexts which makes the scheme fully homomorphic.

As an initial construction, Gentry used ideals and rings without lattices to design the homomorphic encryption scheme, where an ideal is a property preserving subset of the rings such as even numbers. Then, each ideals used in his scheme was represented by the lattices. For example, an ideal I in $\mathbb{Z}[x]/(f(x))$ with $f(x)$ of degree n in an ideal lattice can easily be represented by a column of lattice with basis B_I of length n . Since the bases B_I will produce an $n \times n$ matrix. Gentry's SWHE scheme using ideals and rings is described below:

— **KeyGen Algorithm:** For the given ring R and the basis B_I of ideal I , $IdealGen(R, B_I)$ algorithm generates the pair of (B_J^{sk}, B_J^{pk}) , where $IdealGen()$ is an algorithm outputting the relatively prime public and the secret key bases of the ideal lattice with basis B_I such that $I + J = R$. A $Samp()$ algorithm is also used in key generation to sample from the given coset of the ideal, where a coset is obtained by shifting an ideal by a certain amount. Finally, the public key consists of $(R, B_I, B_J^{pk}, Samp())$ and the secret key only includes B_J^{sk} .

— **Encryption Algorithm:**

For randomly chosen vectors \vec{r} and \vec{g} , using the public key (basis) B_{pk} chosen from one of the "bad" bases of the ideal lattice L , the message $\vec{m} \in \{0, 1\}^n$ is encrypted by:

$$\vec{c} = E(\vec{m}) = \vec{m} + \vec{r} \cdot B_I + \vec{g} \cdot B_J^{pk}, \quad (24)$$

where B_I is basis of the ideal lattice L . Here, $\vec{m} + \vec{r} \cdot B_I$ is called "noise" parameter.

— **Decryption Algorithm:**

By using the secret key (basis) B_J^{sk} , the ciphertext is decrypted as follows:

$$\vec{m} = \vec{c} - B_J^{sk} \cdot \lfloor (B_J^{sk})^{-1} \cdot \vec{c} \rfloor \mod B_I, \quad (25)$$

where $\lfloor \cdot \rfloor$ is the *nearest integer function* which returns the nearest integers for the coefficients of the vector.

- *Homomorphism over Addition:* For the plaintext vectors $\vec{m}_1, \vec{m}_2 \in \{0, 1\}^n$, additive and multiplicative homomorphisms can be verified easily as follows:

$$\vec{c}_1 + \vec{c}_2 = E(\vec{m}_1) + E(\vec{m}_2) = \vec{m}_1 + \vec{m}_2 + (\vec{r}_1 + \vec{r}_2) \cdot B_I + (\vec{g}_1 + \vec{g}_2) \cdot B_J^{pk} \quad (26)$$

It is clear that $\vec{c}_1 + \vec{c}_2$ still preserves the format and is within the ciphertext space. And, to decrypt the sum of the ciphertext, one computes $(\vec{c}_1 + \vec{c}_2) \mod B_J^{pk}$ which is equal to $\vec{m}_1 + \vec{m}_2 + (\vec{r}_1 + \vec{r}_2) \cdot B_I$ for the ciphertexts whose noise amount is smaller than $B_J^{pk}/2$. Then the decryption algorithm works properly and recover the sum of the message $m_1 + m_2$ correctly by taking the modulo B_I of the noise.

- *Homomorphism over Multiplication:* Similarly for the multiplication, after setting $\vec{e} = \vec{m} + \vec{r} \cdot B_I$, the homomorphic property can be expressed as follows:

$$\vec{c}_1 \times \vec{c}_2 = E(\vec{m}_1) \times E(\vec{m}_2) = \vec{e}_1 \times \vec{e}_2 + (\vec{e}_1 \times \vec{g}_2 + \vec{e}_2 \times \vec{g}_1 + \vec{g}_1 \times \vec{g}_2) \cdot B_J^{pk} \quad (27)$$

where $\vec{e}_1 \times \vec{e}_2 = \vec{m}_1 \times \vec{m}_2 + (\vec{m}_1 \times \vec{r}_2 + \vec{m}_2 \times \vec{r}_1 + \vec{r}_1 \times \vec{r}_2) \cdot B_I$. It can be easily verified that the multiplication operation on ciphertexts yields the output still within the ciphertext space. It is said that if the noise $|\vec{e}_1 \times \vec{e}_2|$ is enough small enough the multiplication of plaintexts $\vec{m}_1 \times \vec{m}_2$ can be correctly recovered from the multiplication of ciphertexts $\vec{c}_1 \times \vec{c}_2$.

As seen above, homomorphic operations can be applied to only ciphertexts with small amount of noises. If the noise parameter is very close to a lattice point, further addition and multiplication operations are still allowed. However, after a threshold point, aka critical point, it is not possible to decrypt the ciphertext properly. And, the noise parameter grows linearly with each addition and exponentially with each multiplication operation. This is why the scheme is called Somewhat Homomorphic "for now" allowing only limited number of operations. Since the noise grows much faster with the multiplication operations, the number of multiplication operations before exceeding the threshold is more limited. In order to make the scheme fully homomorphic, the bootstrapping technique was introduced by Gentry. However, the bootstrapping process can be applied to the bootstrappable ciphertexts, which are noisy and have small circuit depth. The depth of the circuit is related to the maximum number of operations. Hence, first the circuit depth is reduced with *squashing* to the degree that the decryption can handle properly.

Squashing: Gentry's bootstrapping technique is allowed only for the decryption algorithms with small depth. Therefore, he used some "tweaks" to reduce the decryption algorithm's complexity. This method is called *squashing* and works as follows:

First, choose a set of vectors, whose sum equals to the multiplicative inverse of the secret key $((B_J^{sk})^{-1})$. If the ciphertext is multiplied by the elements of this set, the polynomial degree of the circuit is reduced to the level that the scheme can handle. The ciphertext is now "bootstrappable". Nonetheless, the hardness of the recovering the secret key is now based on the assumption of Sparse Subset Sum Problem (SSSP) [Hoffstein et al. 2008]. This basically adds another assumption to the provable security of the scheme.

Bootstrapping: Bootstrapping is basically "reencrypting" procedure to get a "fresh" ciphertext from the noisy ciphertext corresponding to the same plaintext. A scheme

is called *bootstrappable* if it can evaluate its own decryption algorithm circuit [Gentry 2009]. First, the ciphertext is transformed into a bootstrappable ciphertext using squashing. Then, by applying bootstrapping procedure, one gets a "fresh" ciphertext. The bootstrapping works as follows: First, it is assumed that two different public and secret key pairs are generated, $(pk1, sk1)$ and $(pk2, sk2)$ and while the secret keys are kept by the client, the public keys are shared with the server. Then, the encryption of the secret key, $Enc_{pk1}(sk1)$, is also transmitted to the server, which already has $c = Enc_{pk1}(m)$. Since the above obtained SWHE scheme can evaluate its own decryption algorithm homomorphically, the noisy ciphertext is decrypted homomorphically using $Enc_{pk1}(sk1)$. Then, the result is encrypted using a different public key $pk2$, i.e., $Enc_{pk2}(Dec_{sk1}(c)) = Enc_{pk2}(m)$. Since the scheme is assumed semantically secure, an adversary can not distinguish the encryption of secret key from the encryption of 0. The last ciphertext can be decrypted using $sk2$, which is kept secret by the client, i.e., $Dec_{sk2}(Enc_{pk2}(m)) = m$. In brief, first the homomorphic decryption of the noisy ciphertext removes the noise, and then the new homomorphic encryption introduces new small noise to the ciphertext. Now, the ciphertext is like just encrypted. Further homomorphic operations can be computed on this "fresh" ciphertext until reaching again to a threshold point. Note that Gentry's bootstrapping method increases the computational cost noticeably and becomes a major drawback for the practicality of FHE. In a nutshell, starting from constructing a SWHE scheme and then squashing method to reduce the circuit depth of decryption algorithm and the bootstrapping to obtain fresh ciphertext completes the creation of a FHE scheme. Hence, one can apply bootstrapping repetitively to compute unlimited number of operations on the ciphertexts to successfully have an FHE scheme.

After Gentry's original scheme, some of the follow-up works tried to generally improve Gentry's original work. In [Gentry 2009], Gentry's key generation algorithm is used for a particular purpose only and the generation of an ideal lattice with a "good" basis is left without a solution. Gentry introduced a new *KeyGen* algorithm in [Gentry 2010] and improved the security of the hardness assumption of SSSP by presenting a quantum worst case/average case reduction. However, a more aggressive analysis of the security of SSSP was completed by Stehle and Steinfeld [Stehlé and Steinfeld 2010]. They also suggested a new probabilistic decryption algorithm with lower multiplicative degree, which is square root of previous decryption circuit degree. Moreover, a new FHE scheme, which was a variant of Gentry's scheme was introduced in [Smart and Vercauteren 2010]. The scheme uses smaller ciphertext and key sizes than Gentry's scheme without sacrificing the security. Some later works [Gentry and Halevi 2011; Scholl and Smart 2011; Ogura et al. 2010] focused on the optimizations in the key generation algorithm in order to implement the FHE efficiently. Moreover, Mikuš proposed a new SWHE scheme with bigger plaintext space to improve the number of homomorphic operations with slight increase in complexity of the key generation algorithm [Mikuš 2012].

3.3.2. FHE schemes Over Integers. In 2010, one year after Gentry's original scheme, another SWHE scheme is presented in [Van Dijk et al. 2010] which suggests Gentry's ingenious bootstrapping method in order to obtain an FHE scheme. The proposed scheme is over integers and the hardness of the scheme is based on the *Approximate-Greatest Common Divisor* (AGCD) problems [Galbraith et al. 2016]. AGCD problems try to recover p from the given set of $x_i = pq_i + r_i$. The primary motivation behind the scheme is its conceptual simplicity. A symmetric version of the scheme is probably one of the simplest schemes. The proposed symmetric SWHE scheme is described as follows:

- *KeyGen Algorithm*: For the given security parameter λ , a random odd integer p of bit length η is generated.
- *Encryption Algorithm*: For a random large prime numbers p and q , choose a small number $r \ll p$. Then, the message $m \in \{0, 1\}$ is encrypted by:

$$c = E(m) = m + 2r + pq, \quad (28)$$

where p is kept hidden as private key and c is the ciphertext.

- *Decryption Algorithm*: The ciphertext can be decrypted as follows:

$$m = D(c) = (c \bmod p) \bmod 2. \quad (29)$$

Decryption works properly only if $m + 2r < p/2$. This actually restricts the depth of the homomorphic operations performed on the ciphertext. Then, Dijk et al. used Gentry's squashing and bootstrapping techniques to make the scheme fully homomorphic. The homomorphic properties of the scheme can be shown easily as follows:

- *Homomorphism over addition*:

$$E(m_1) + E(m_2) = m_1 + 2r_1 + pq_1 + m_2 + 2r_2 + pq_2 = (m_1 + m_2) + 2(r_1 + r_2) + (q_1 + q_2)p. \quad (30)$$

The output clearly falls within the ciphertext space and can be decrypted if the noise $|m_1 + 2r_1 + m_2 + 2r_2| < p/2$, where p is the private key. Since $r_1, r_2 \ll p$, various number of additions can still be performed on ciphertext before noise exceeds $p/2$.

- *Homomorphism over Multiplication*:

$$E(m_1)E(m_2) = (m_1 + 2r_1 + pq_1)(m_2 + 2r_2 + pq_2) = m_1m_2 + 2(m_1r_2 + m_2r_1 + 2r_1r_2) + kp. \quad (31)$$

The output preserves the format of original ciphertexts and holds the homomorphic property. The encrypted data can be decrypted if the noise is smaller than half of the private key, i.e., $|m_1m_2 + 2(m_1r_2 + m_2r_1 + 2r_1r_2)| < p/2$. The noise grows exponentially with the multiplication operation. This puts more restriction over homomorphic multiplication operation than addition.

In fact, the scheme presented so far [Van Dijk et al. 2010] was the symmetric version of the homomorphic encryption. Transforming the underlying symmetric HE scheme into an asymmetric HE scheme is also presented in [Van Dijk et al. 2010]. It is enough to compute many "encryptions of zero" $x_i = pq_i + 2r_i$, where p is private key. Then, many x_i s are shared as the public key. To encrypt the message with public key, it is enough to add the message to a subset sum of x_i s. Same decryption is used to decrypt the ciphertext. As there is no efficient algorithm to recover p from the given x_i s in polynomial time, the scheme is considered as secure. The scheme is now basically a public key encryption scheme, since it uses different keys to encrypt and decrypt.

The FHE scheme proposed in [Van Dijk et al. 2010] is conceptually very simple. However, this simplicity comes with a cost in computations. So, the scheme is not very efficient. Hence, some early attempts directly tried to improve the efficiency. For example, some follow-up optimizations focused on reducing the size of public keys [Coron et al. 2011] ($O(\lambda^{10}) \rightarrow O(\lambda^7)$), [Coron et al. 2012] ($O(\lambda^7) \rightarrow O(\lambda^5)$), [Yang et al. 2012] ($O(\lambda^5) \rightarrow O(\lambda^3)$). A more efficient public key generation [Ramaiah and Kumari 2012b] and re-encryption [Chen et al. 2014] are other suggested works without reducing the security of the scheme. Later, an important variant, which is batch FHE over integers, was proposed [Cheon et al. 2013] (merged version of [Coron et al. 2013] and [Kim et al. 2013]). Batch FHE has the ability to pack multiple ciphertexts into a single ciphertext. Moreover, the proposed scheme provides two options for the hardness of the base problem: Decisional AGCD and Error-free AGCD. In [Cheon et al. 2013], it is also shown how to achieve decryption operation in parallel l -slots.

Some further approaches for FHE schemes over integers are also proposed: a new scale invariant FHE over integers [Coron et al. 2014], a new scheme with integer plaintexts [Ramaiah and Kumari 2012a], a new SWHE scheme for computing arithmetic operations on large integer numbers without converting them into bits [Pisa et al. 2012], a new symmetric FHE without bootstrapping [Aggarwal et al. 2014], and a new FHE for non-binary message spaces [Nuida and Kurosawa 2015]. All these schemes improved FHEs over integers in the way that their names imply.

3.3.3. LWE-based FHE schemes. Learning with Error (LWE) is considered as one of the hardest problems to solve in practical time for even post-quantum algorithms. First, it was introduced by Oded Regev as an extension of "learning from parity with error" problem [Regev 2009]. Regev reduced the hardness of worst-case lattice problems like SVP to LWE problems, which means that if one can find an algorithm that can solve LWE problem in an efficient time, the same algorithm will also solve the SVP problem in an efficient time. Since then, it is one of most attractive and promising topics for post-quantum cryptology with its relatively small ciphertext size. Lyubashevsky et al. suggested another significant improvement on the LWE problem which may lead to a new applications by introducing ring-LWE (RLWE) problem [Lyubashevsky et al. 2013]. The RLWE problem is an algebraic variant of LWE, which is more efficient for practical applications with strong security proofs. They proved that the RLWE problems are reducible to worst-case problems on ideal lattices, which is hard for polynomial-time quantum algorithms.

In the LWE-based FHE schemes, an important step towards to a practical FHE scheme is made in [Brakerski and Vaikuntanathan 2011]. Brakerski and Vaikuntanathan established a new SWHE scheme based on Ring-Learning with Error (RLWE) to take advantage from the efficiency feature of RLWE [Brakerski and Vaikuntanathan 2011]. In other words, although both LWE and RLWE problems can be used as the hardness assumption of an FHE scheme, RLWE shows better performance. Then, the scheme uses Gentry's blueprint squashing and bootstrapping techniques to obtain an FHE scheme. They used polynomial-LWE (PLWE), which is simplified version of RLWE. PLWE is also reducible to worst-case problems such as SVP on ideal lattices. The schemes proposed after [Brakerski and Vaikuntanathan 2011] is also called second generation FHE schemes.

Below, for the sake of simplicity, as we did in the previous part, we first show symmetric version of the scheme and then how to transform the scheme into a asymmetric HE scheme.

Notation: A very common notation is that $\langle a, b \rangle$ is used to denote the inner product of vectors a and b . Moreover, $d \xleftarrow{\$} \mathcal{D}$ denotes that d is randomly assigned by an element from the distribution \mathcal{D} and $\mathbb{Z}[x]/(f(x))$ denotes the ring of all polynomials modulo $f(x)$. The ring of polynomials modulo $f(x)$ with coefficients in \mathbb{Z}_q is denoted with $R_q \equiv \mathbb{Z}_q[x]/(f(x))$. Finally, χ denotes an error distribution over the ring R_q .

Symmetric version of the underlying scheme is given as follows:

- *KeyGen Algorithm:* An element of the ring is chosen as a secret key from the error distribution, i.e., $s \xleftarrow{\$} \chi$. Then, the secret key vector is described as $\vec{s} = (1, s, s^2, \dots, s^D)$ for an integer D .
- *Encryption Algorithm:* After choosing a random vector $a \xleftarrow{\$} R_q^n$ and the noise $e \xleftarrow{\$} \chi$, the message m is encrypted by:

$$\vec{c} = (c_0, c_1) = (as + te + m, -a) \quad (32)$$

where $\vec{c} \in R_q^2$.

- Decryption Algorithm:* In order to decrypt the ciphertext to recover the message, it can be easily computed that:

$$m = \langle \vec{c}, \vec{s} \rangle \pmod{t}. \quad (33)$$

Decryption works properly if $\langle \vec{c}, \vec{s} \rangle$ is smaller than $q/2$. Furthermore, in order to make the scheme asymmetric, it is sufficient to generate a random set of pairs $(a, as + te)$. Also, the homomorphic property of the scheme is very similar to those in [Gentry 2009] and [Van Dijk et al. 2010].

- Homomorphism over Addition:*

$$E(m) + E(m') = (c_0 + c'_0, c_1 + c'_1) = ((a + a')s + t(e + e') + (m + m'), -(a + a')), \quad (34)$$

Similar to previous schemes, decryption works if the noise is small. And, it is clear that homomorphically added ciphertexts keep the format of the original ciphertexts and stay within the ciphertext space.

- Homomorphism over Multiplication:*

$$E(m) + E(m') = (c_0 c'_0, c_1 c'_1) = (-a' s^2 + (c'_0 a + c_0 a')s + t(2ee' + em' + e'm) + mm'). \quad (35)$$

The output seems almost like a ciphertext, but it still can be decrypted correctly with the expense of a new cost by adding a new term to ciphertext.

Brakerski and Vaikuntanathan made their scheme fully homomorphic using Gentry's blueprint squashing and bootstrapping. They also showed their SWHE scheme is circular secure (aka Key-Dependent message (KDM) security) with respect to linear functions of secret key, i.e., the encryption can successfully keep secure linear functions of its own secret key.

After the proposed BGN-type cryptosystem based on LWE, which is additively homomorphic and allowing only one multiplication operation in [Gentry et al. 2010], Brakerski and Vaikuntanathan proposed another SWHE scheme based on standard LWE problems using *re-linearization* technique [Brakerski and Vaikuntanathan 2014a]. Re-linearization makes the long ciphertexts, which are the output of the homomorphic evaluation, regular size. Another important contribution in this work is the dimension-modulus reduction, which does not require an SSSP assumption and squashing method used in Gentry's original framework.

As discussed earlier, Gentry's bootstrapping method is a creative method to obtain an FHE scheme, however it comes with a huge cost. A *leveled*-FHE scheme without using the bootstrapping technique was introduced by [Brakerski et al. 2012]. Levelled FHE can evaluate homomorphic operations for only a predetermined circuit depth level. Brakerski et al. [Brakerski et al. 2012] also showed that their scheme with bootstrapping still provides better performance than the one without bootstrapping and also suggested the batching as an optimization. To achieve batching, "modulus switching" technique is used iteratively to keep the noise size constant. Then, Brakerski removed the necessity of modulus switching in [Brakerski 2012]. In Brakerski's new scale invariant FHE scheme [Brakerski 2012], contrary to the existing FHE schemes, the noise grows linearly with the evaluation of homomorphic operations instead of exponentially and the scheme is based on the hardness of *GapSVP problem* [Peikert 2015]. GapSVP problem is roughly deciding the existence of a shorter vector than the vector with length d for a given lattice basis B . The result returns simply yes or no. Then, Fan and Vercauteren optimized the Brakerski's scheme by changing the based assumption to RLWE problem [Fan and Vercauteren 2012]. Some other modifications to [Brakerski 2012] focused on reducing the overhead of key switching and faster evaluation of homomorphic operations [Wu et al. 2012] and using re-linearization to improve efficiency [Zhang et al. 2014].

Recently, by [Gentry et al. 2013] a significant FHE scheme was introduced claiming three important properties: simpler, faster, and attribute-based FHE. The scheme is simpler and faster due to the "approximate eigenvector" method replacing the re-linearization technique. In this method, by keeping only some parameters small, the format of the ciphertext can be preserved under the evaluation of homomorphic operations. In the previous schemes which uses the bootstrapping technique, the secret key (evaluation key) of the user is sent to the cloud to evaluate the ciphertext homomorphically for the bootstrapping. In contrast, [Gentry et al. 2013] eliminates that need and leads to propose the first identity-based FHE scheme, which allows homomorphic evaluation by only a target identity having the public parameters. Then, Brakerski and Vaikuntanathan followed [Gentry et al. 2013] to construct an FHE scheme secure under a polynomial LWE assumption [Brakerski and Vaikuntanathan 2014b]. It is shown that the proposed scheme is as secure as any other lattice-based PKE scheme. Recently, Painsavoin and Vialla showed a way of minimizing the number of required bootstrapping based on the linear programming techniques that can be applied to [Gentry et al. 2013] as well.

In addition to more recently proposed LWE-based FHE schemes in [Zhang et al. 2014; Chen et al. 2014; Tanping et al. 2015; Wang et al. 2015b], some optimizations focused on better (faster) bootstrapping algorithms [Alperin-Sheriff and Peikert 2013; Alperin-Sheriff and Peikert 2014], speeding homomorphic operations [Gentry et al. 2012], and a new extension to FHE for multi-identity and multi-key usage [Clear and McGoldrick]. More recently, a new efficient SWHE scheme based on the polynomial approximate common divisor problem is presented in [Cheon et al. 2016]. The presented scheme in [Cheon et al. 2016] can handle efficiently large message spaces.

3.3.4. NTRU-like FHE schemes. To obtain a practical and applicable FHE scheme, one of the crucial steps is taken by showing the construction of an FHE scheme from NTRU-Encrypt, which is an old encryption scheme proposed by Hoffstein, Pipher, and Silverman in [Hoffstein et al. 1998]. Specifically, how to obtain a multi-key FHE from the NTRU-Encrypt (called NTRU) was shown by [López-Alt et al. 2012]. NTRU encryption scheme is one of earliest attempts based on lattice problems. Compared with RSA and GGH cryptosystems, NTRU improves the efficiency significantly in both hardware and software implementations. However, there were security concerns for 15 years until the study done by [Stehlé and Steinfeld 2011]. They reduced the security of the scheme to standard worst-case problems over ideal lattices by modifying the key generation algorithm. Since the security of the scheme is improved, its high efficiency, easy implementation, and standardization features again attract researchers' interest. At the same time, [López-Alt et al. 2012] and [Gentry 2012] independently noticed the fully homomorphic properties of the NTRU encryption. López-Alt et al. used the NTRU encryption scheme to obtain a practical FHE [López-Alt et al. 2012] with three differences. First, the set from which the noise is sampled is changed from a deterministic set to a distribution. Second, the modification introduced in [Stehlé and Steinfeld 2011], which makes the scheme more secure, is used and third, the parameters are chosen to allow fully homomorphism. Their proposed NTRU-like encryption scheme in [López-Alt et al. 2012] is as follows:

- *KeyGen Algorithm:* For chosen sampled polynomials f' and g from a distribution χ (specifically, a discrete Gaussian distribution), it is set $f = 2f' + 1$ to get $f \equiv 1 \pmod{2}$ and f is invertible. Then, the secret key $sk = f \in R$ and public key $pk := h = 2gf^{-1} \in R_q$.

- *Encryption Algorithm:* For chosen samples s and e from the same distribution χ , the message m is encrypted by:

$$c = E(m) = hs + 2e + m, \quad (36)$$

where the ciphertext $c \in R_q$.

- *Decryption Algorithm:* The ciphertext can easily be decrypted as follows:

$$m = D(c) = fc \pmod{2}, \quad (37)$$

where $fc \in R_q$. The correctness of the scheme can be verified using $h = 2gf^{-1}$ and $f \equiv 1 \pmod{2}$. Moreover, the scheme proposed by López-Alt et al. is a new type of FHE scheme, which is called multi-key FHE. Multi-key FHE has ability to evaluate on ciphertexts which are encrypted with independent keys, i.e., each user can encrypt data with her own public key and a third party can still perform homomorphic evaluation on these ciphertexts. The only interaction required between the users is to obtain a "joint secret key". The homomorphically evaluated ciphertext is decrypted by using the joint secret key, which is obtained by using all involved secret keys. The message m_i is encrypted by using public key $h_i = 2g_i f_i^{-1}$ with the formula, $c_i = h_i s_i + 2e_i + m_i$. The multikey homomorphism properties for two party computation is shown using joint secret key $f_1 f_2$.

- *Multi-key Homomorphism over Addition:*

$$\begin{aligned} f_1 f_2 (c_1 + c_2) &= 2(f_1 f_2 e_1 + f_1 f_2 e_2 + f_2 g_1 s_1 + f_1 g_2 s_2) + f_1 f_2 (m_1 + m_2) \\ &= 2e_{add} + f_1 f_2 (m_1 + m_2) \end{aligned} \quad (38)$$

- *Multi-key Homomorphism over Multiplication:*

$$\begin{aligned} f_1 f_2 (c_1 c_2) &= 2(2g_1 g_2 s_1 s_2 + g_1 s_1 f_2 (2e_2 + m_2) + g_2 s_2 f_1 (2e_1 + m_1) \\ &\quad + f_1 f_2 (e_1 m_2 + e_2 m_1 + 2e_1 e_2)) + f_1 f_2 (m_1 m_2) \\ &= 2e_{mult} + f_1 f_2 (m_1 m_2) \end{aligned} \quad (39)$$

Here, it is seen that multi-key homomorphic operation increases noise more than a single key homomorphic evaluation. However, $m_1 + m_2$ and $m_1 m_2$ can still be recovered correctly using the jointly obtained secret key since f, g, s, e all are sampled from the bounded distribution χ . In other words, the decryption still works if the each of the noise parameters e_{add} and e_{mult} are smaller than $|p/2|$.

As observed in all of the FHE schemes presented in detail in our work, since in [López-Alt et al. 2012] noise grows with homomorphic operations on encrypted data, the proposed scheme is actually an SWHE scheme. To make it fully homomorphic, López-Alt et al. also (like all others above) used Gentry's bootstrapping technique. However, to apply bootstrapping, one first needs to make the underlying SWHE scheme bootstrappable. For this reason, first modulus reduction technique described in [Brakerski 2012; Brakerski and Vaikuntanathan 2014a] was used. Then, the final scheme was named a leveled-FHE because it had the ability to deal only a limited number of public keys. Although the number of parties that can be used in homomorphic operations is limited, the complexity of circuit that can be used in homomorphic operations is still independent from the number of party that can join the communication.

Another issue to be taken account in [López-Alt et al. 2012] is the assumptions. Specifically, two assumptions are used in the scheme proposed by Lopez-Alt et al. The first is RLWE problems and the second is Decisional Small Polynomial Ratio (DSPR). Though RLWE is well-studied and about being a standard problem, DSPR assumption is a non-standard one. Hence, in [Bos et al. 2013], Bos et al. showed how to modify

Table III: "Fully" implemented FHE schemes

Scheme Information		Platform	Parameters		Running Times				
Implemented Scheme	Base Scheme	Software	Security parameter, λ	dimension, n	PK size	KeyGen	Enc	Dec	Recrypt
GH11 [Gentry and Halevi 2011]	Gen09 [Gentry 2009]	C/C++	72	33768	2.25 GB	2.2 h	3 min (SWHE)	0.66 s (SWHE)	31 min
CMNT11 [Coron et al. 2011]	DGHV10 [Van Dijk et al. 2010]	Sage 4.5.3	72	7897	802 MB	43 min	2 min 57 s	0.05 s	14 min 33 s
CNT12 (with compressed PK) [Coron et al. 2012]	DGHV10 [Van Dijk et al. 2010]	Sage 4.7.2	72	7897	10.3 MB	10 min	7 min 15 s	0.05 s	11 min 34 s
CNT12 (leveled) [Coron et al. 2012]	DGHV10 [Van Dijk et al. 2010]	Sage 4.7.2	72	5700	18 MB	6 min 18 s	3.4 s	0.00 s	2 h 27 min

[López-Alt et al. 2012] to remove DSRP assumption. While removing DSRP assumption, the *tensoring* technique introduced in [Brakerski 2012] is used to restrict the noise. The new presented scheme in [Bos et al. 2013] is also scale invariant and a practical variant of their scheme is also presented as another contribution. Furthermore, in [Bos et al. 2013], it is also shown how to use larger inputs via Chinese Remainder theorem (CRT).

Two follow-up interesting works improved the NTRU-like FHE using different techniques. While one of them focuses on a customized and a generic bit-sliced implementation of NTRU-like FHE schemes [Doröz et al. 2014] and the other suggests the use of GPU [Dai et al. 2014]. Furthermore, in [Doröz et al. 2014], the AES circuit is chosen to evaluate the homomorphic operations, which is faster than the proposed one in [Gentry et al. 2012]. Other improvements on hardware implementations of NTRU-like FHE schemes are more recently published in [Liu and Wu 2015; Doröz et al. 2015]. Another NTRU-like FHE scheme was suggested in [Rohloff and Cousins 2014]. They used the bootstrapping proposed in [Alperin-Sheriff and Peikert 2013] and "double-CRT" proposed in [Gentry et al. 2012] to modify the representation of the ciphertexts in more efficient way.

4. IMPLEMENTATIONS OF SWHE AND FHE SCHEMES

The ultimate goal with different HE schemes is to obtain an unbounded and practical FHE scheme. PHE schemes and SWHE schemes proposed before Gentry's breakthrough work in 2009 were stepping stone towards that goal. Nonetheless, they are restricted in terms of the areas that can be applied. However, the SWHE schemes proposed after Gentry's work are mostly the part of the FHE schemes rather than a different scheme. Moreover, a bounded (level) FHE can also be called as SWHE scheme. Hence, it is not possible to separate SWHE and FHE schemes for the works proposed after Gentry's work. In this section, we aim to summarize the implementations of the SWHE and FHE schemes, which can lead to the new works and speed up the follow-up works, proposed after Gentry's work.

Implementation of a cryptographic scheme is the middle step between designing the scheme and applying it to a real life service and it provides a realistic performance assessment of the designed scheme. Although some new proposed FHE schemes have increased the efficiency and performance of the implementations significantly, the overhead and cost of the FHE implementations are still too high to be applied transparently in a real life service without disturbing the user.

After solving the long term open problem of designing a fully homomorphic scheme [Gentry 2009], many new fully homomorphic scheme proposals were tested with implementation. In a very first attempt, Smart and Vercauteren implemented their scheme in [Smart and Vercauteren 2010], which is a variant of Gentry's original scheme. However, their key generation takes hours up to $N = 2^{11}$, where N is the lattice dimension and does not generate the key pairs after $N = 2^{11}$. More importantly, their imple-

Table IV: FHE implementations for "Low-depth" circuits

Scheme Information		Platform	Parameters		Running times			
Implemented Scheme	Base Scheme	Software			Enc	Dec	Mult	Add
NLV11 [Naehrig et al. 2011]	BV11 [Brakerski and Vaikuntanathan 2011]	Magma	$w = 2^{32}$	$q=127$	756 ms	57 ms	1590 ms	4 ms
YASHE (by BLLN13 [Bos et al. 2013])	LTV12 [López-Alt et al. 2012]	C/C++	$t = 2^{10}$	$q=130$	27 ms	5 ms	31 ms	0.024 ms
YASHE (by LN14a [Lepoint and Naehrig 2014])	LTV12 [López-Alt et al. 2012]	C/C++	$w = 2^{32}$	$q=130$	16 ms	15 ms	18 ms	0.7 ms
FV (by LN14a [Lepoint and Naehrig 2014])	BV11 [Brakerski and Vaikuntanathan 2011]	C/C++	$w = 2^{32}$	$q=130$	34 ms	16 ms	59 ms	1.4 ms
RC14 [Rohloff and Cousins 2014]	LTV12 [López-Alt et al. 2012]	Matlab	$n = 2^{10}$	$t=1$	12 ms	3.36 ms	100 ms	0.56 ms

mentation did not include the bootstrapping procedure. Hence, it is actually a SWHE scheme as it was implemented. Then, Craig Gentry and Shai Halevi [Gentry and Halevi 2011] succeeded to implement the FHE scheme first time by continuing the way that Smart and Vercauteren had started. The running times for the implementation in [23] and other proposed FHE implementations which are evaluated over random depth circuits is given in Table III. Moreover, Gentry and Halevi in [Gentry and Halevi 2011] introduced some optimizations and simplifications on the squashing process to obtain a bootstrappable scheme. In their implementation, they showed four security levels: toy, small, medium, and large. They suggested that the large parameter settings are practically secure, which have a lattice dimension of 2^{15} . However, the performance of the implementation is very inefficient in practical terms. For the large parameter setting, a key pair was generated at 2.2 hours and public key size was 2.25 GB. Re-encrypting the ciphertexts (bootstrapping) took 31 minutes. After that, in [Coron et al. 2011], an integer variant of the FHE scheme introduced originally in [Van Dijk et al. 2010] was implemented. In this implementation, the key generation takes 43 min, and the public key size is 802 MB. The implementation showed that the same security level can be achieved with a much simpler scheme. (The difference comes from the different definitions of security levels). Later, Coron et al. in a different work [Coron et al. 2012] improved public key size to 10 MB, key generation to 10 minutes, and re-encryption procedure to 11 min 34 seconds using the similar parameter settings in [Coron et al. 2011]. This performance is obtained using a compression technique on the public key. In [Coron et al. 2012], a levelled DGHV scheme is also implemented with slightly worse performance. Yuanmi Chen and Phong Q. Nguyen proposed an algorithm to break the scheme in [Coron et al. 2012], which is faster than exhaustive search. This work showed that the security level of the scheme proposed in [Coron et al. 2012] is much lower than the scheme proposed in [Gentry and Halevi 2011].

The second type of FHE implementations tried to implement leveled-FHE schemes for small depth circuits with given run time for isolated and composed addition and multiplication [Naehrig et al. 2011; Bos et al. 2013; Lepoint and Naehrig 2014; Rohloff and Cousins 2014]. The comparisons for these small-depth FHE implementations are given at Table IV. Since the performance of the state of the art was unsatisfactory, as an early attempt, a relatively simpler FHE, which allows only a few homomorphic multiplication operations was implemented in [Naehrig et al. 2011]. Later, this performance was improved by Bos et al. [Bos et al. 2013] due to the new method to evaluate the homomorphic multiplication operation. Moreover, unlike [Naehrig et al. 2011], in [Bos et al. 2013] the underlying scheme was implemented in C programming language to avoid the unwelcome overhead due to the computer algebra system. Then, a similar performances with [Bos et al. 2013] is obtained. Recently, a significant improvement is

Table V: Comparison of "real world" complex FHE implementations

Scheme			Used Memory	Parameters		Running Times		
Implemented Scheme	Base scheme	Circuit	RAM	λ	AND depth	total evaluation time	number of parallel enc	relative time
GHS12 (original)(packed) [Gentry et al. 2012]	BGV11 [Brakerski et al. 2011]	AES	256 GB	80	40	48 hours	54	37 min
GHS12 (original)(byte-sliced) [Gentry et al. 2012]						65 hours	720	5 min
CLT13 (byte-wise) [Cheon et al. 2013]	DGHV10 [Van Dijk et al. 2010]	AES	32 GB	72	40	18.3 hours	33	33 min
CLT13 (state-wise) [Cheon et al. 2013]						113 hours	531	12 min 46 s
CLT14 (state-wise) [Coron et al. 2014]	DGHV10 [Van Dijk et al. 2010]	AES	*	80	40	102 hours	1875	3 min 15 s
CLT14 (state-wise) [Coron et al. 2014]				72		3 h 35 min	569	23 s
LN14a (YASHE) [Lepoint and Naehrig 2014]	LTV12 [López-Alt et al. 2012]	SIMON	*	128	34	1 h 10 min	2048	2.04 s
LN14a (FV) [Lepoint and Naehrig 2014]	Bra12 [Brakerski 2012]					3 h 27 min	2048	6.06 s
DHS14 [Doröz et al. 2014]	LTV12 [López-Alt et al. 2012]	AES	*	~80	40	31 hours	2048	55 s
DSES14 [Doröz et al. 2014]	Prince	32 GB	130	30		57 min	1024	3.3 s
ARSTZ15 [Albrecht et al. 2015]	BGV11 [Brakerski et al. 2011]	LowMC	16 GB	80	12	8 min	600	0.8 s
GHS12 (updated)(no bootstrapping) [Gentry et al. 2012]	BGV11 [Brakerski et al. 2011]	AES	4 GB	80	40	4 min 12 s	120	2 s
GHS12 (updated)(with bootstrapping) [Gentry et al. 2012]						17 min 30 s	180	5.8 s

made by using double-CRT in the representation of ciphertexts and used parallelism to accelerate the implementation in Matlab [Rohloff and Cousins 2014].

In contrast to above schemes, which are either proof of concept or small-depth implementations, the authors in [Gentry et al. 2012] implemented FHE for the first time to evaluate circuit complex enough for a real life application. In [Gentry et al. 2012] Gentry et al. implemented a variant of BGV scheme proposed in [Brakerski et al. 2011], which is a leveled FHE without bootstrapping, in order to evaluate AES circuit homomorphically. Actually, the idea of homomorphic evaluation of AES is first discussed in [Naehrig et al. 2011] with the following scenario. A client first sends the key of AES by encrypting with FHE, $FHE(K)$. Then, the client uploads the data by encrypting with AES only, $AES_K(m)$. When the cloud wants to evaluate the data homomorphically, it computes $FHE(AES_K(m))$ and decrypts AES homomorphically (blindfold) to obtain $FHE(m)$. After that, the cloud can compute every homomorphic operation on the data encrypted with FHE. The comparison of such more complex "real world" FHE implementations are presented in Table V. A realization of how to achieve SIMD (single-instruction multiple-data) operations using homomorphic evaluation of AES is proposed by Smart and Vercauteren [Smart and Vercauteren 2011]. Later, some works [Coron et al. 2013; Mella and Susella 2013; Coron et al. 2014; Doröz et al. 2014] also improved the performances of the homomorphic evaluation of AES circuit by applying the recent improvements and optimizations in theoretical side. In addition to the use of AES circuit to evaluate homomorphically, lightweight block ciphers such as Prince [Doröz et al. 2014], SIMON [Lepoint and Naehrig 2014], and LowMC [Albrecht et al. 2015] are also proposed. In [Mella and Susella 2013], Mella and Susella estimated the cost of some of the symmetric cryptographic primitives such as AES-128, SHA-256 hash function, Salsa20 stream cipher, and KECCAK sponge function. They concluded that AES is best suited for the homomorphic evaluation because of its low number of rounds and absence of integer operations and logical ANDs in its internals. However, in [Mella and Susella 2013], only AES-128 is implemented.

Although all aforementioned implementations are published in the literature, unfortunately only few of them are publicly available to researchers. Some of the publicly available implementations are listed in Table VI. From publicly available implementations, HELib [Halevi and Shoup 2013b] is the most important and widely utilized one. HELib implements the BGV scheme [Brakerski et al. 2011] with Smart-Vercauteren

Table VI: Some publicly available FHE implementations

Name	Scheme	Language	Documentation	Libraries
HElib [Halevi and Shoup 2013b]	BGV [Brakerski et al. 2011]	C++	Yes [Halevi and Shoup 2013a]	NTL, GMP
libScarab [Perl et al. 2011a]	SV [Smart and Vercauteren 2010]	C	Yes [Perl et al. 2011b]	GMP, FLINT, MPFR, MPIR
FHEW [Ducas and Micciancio 2014]	DM14 [Ducas and Micciancio 2015]	C++	Yes [Ducas and Micciancio 2015]	FFTW

ciphertext packing techniques and some new optimizations. The design and implementation of HElib are documented in [Halevi and Shoup 2013a] and algorithms used in HElib are documented in [Halevi and Shoup 2014]. HElib is designed using low-level programming, which deals with the hardware constraints and components of the computer without using the functions and commands of a programming language and hence, defined as "assembly language for HE". It was implemented using GPL-licensed C++ library. Since December 2014, it supports bootstrapping [Halevi and Shoup 2015] and since March 2015, it supports multi-threading. In an important extension, homomorphic evaluation of AES was implemented on top of HElib [Gentry et al. 2012] and included in the HElib source code in [Halevi and Shoup 2013b].

Unfortunately, the usage of HElib is not easy because of the sophistication needed for its low-level implementation and parameter selection which effects both performance and security level. Another notable open source FHE implementation is libScarab [Perl et al. 2011a]. To the best of our knowledge, libScarab [Perl et al. 2011a] is the first open-source implementation of FHE. Its parameter selection is relatively easier than that of HElib, but it suffers from a lot of limitations. For instance, it does not implement modern techniques (e.g., modulus reduction and re-linearization techniques [Brakerski and Vaikuntanathan 2014a]) to handle the noise level or it also does not support the SIMD techniques introduced in [Smart and Vercauteren 2014]. It implements Smart-Vercauteren's FHE scheme in [Smart and Vercauteren 2010] and documentation is provided in [Perl et al. 2011b].

More recently, another major implementation is introduced by Ducas and Micciancio and called "Fastest Homomorphic Encryption in the West" (FHEW) [Ducas and Micciancio 2014]. It is documented in [Ducas and Micciancio 2015]. It significantly improves the time required to bootstrap the ciphertext claiming homomorphic evaluation of a NAND gate "in less than a second". A NAND gate is functionally complete. Hence, any possible boolean circuits can be built using only NAND gates. In [Ducas and Micciancio 2015], the usage of ciphertext packing and SIMD techniques provides an amortized cost. However, in FHEW such performance is achieved using only a few hundred lines of code with the use of one additional library, FFTW [Frigo and Johnson 2005].

The first known usage of FHE in a production environment is announced by Fujitsu Laboratories Ltd. [Ltd. 2013]. Their reported implementation provides statistical calculations and biometric authentication by using FHE-based security. They improved a FHE by batching the string bits of data. The practical testing of this FHE implementation by Fujitsu is still pending as of this writing. Although the software only implementations are considered promising to obtain a practical FHE implementation, there is still a substantial gap between the achieved and the targeted performance. This gap led to new alternative research area in hardware implementations. The hardware solutions to accelerate both FHE and SWHE schemes mainly focused on three implementation platforms: Graphics Processing Unit (GPU), Application-Specific Integrated Circuit (ASIC), and Field-Programmable Gate Array (FPGA) (A useful survey of hardware implementations of homomorphic schemes can be found in [Moore et al.

2014b]). Although GPU is for graphical purposes, its highly parallel structure offers great promise over CPU for efficiency. Hence, it is suggested in some studies to use GPU order to improve the efficiency of homomorphic evaluation [Dai et al. 2014; Wang et al. 2014; Wang et al. 2015; Dai et al. 2015; Lee et al. 2015]. One of the major barriers to a practical FHE is the noise growth in homomorphic multiplication operation. This prompted researchers to find a solution that can deal with a large number of modular multiplications. Therefore, there are some works focusing particularly on this problem using the customized ASICs [Doroz et al. 2013b; Wang et al. 2014; Doroz et al. 2013a]. In spite of the potential of GPU and ASIC solutions, most of the proposed studies are based on the reconfigurable hardware, specifically FPGA. FPGA platforms offer not only Fast Fourier Transform (FFT), but also some optimization techniques such as number theoretic transformation (NTT) and fast modular polynomial reduction at hardware level. Such large and reconfigurable environment provided by FPGAs motivates many researchers to speed up the practicality of FHE schemes [Cousins et al. 2012; Wang and Huang 2013; Cao et al. 2013; Moore et al. 2013; Chen et al. 2015; Cao et al. 2014; Moore et al. 2014a; Cousins et al. 2014; Roy et al. 2015; Pöppelmann et al. 2015; Öztürk et al. 2015].

In conclusion, some of the SWHE implementations (leveled-FHE) [Gentry et al. 2012] get closer to a tolerable performance. However, the bootstrapping techniques in FHE schemes need to be improved and the cost of homomorphic multiplications should be reduced to increase the performance.

5. FURTHER RESEARCH DIRECTIONS AND LESSONS LEARNED

Performance of any encryption scheme is evaluated with three different criteria: security, speed, and simplicity. First, an encryption scheme must be secure so that an attacker can not obtain any type of information by using reasonable amount of resources. Second, its efficiency must not disturb the user's comfort, i.e., it must be transparent to the users because users prefer usability against security. Lastly, if and only if an encryption scheme is understandable by the other area practitioners, they will implement the scheme for their applications and productions. If the existing FHE schemes are evaluated in terms of the three criteria, there is, though getting closer, still a substantial room for improvement in terms of all these criteria, especially for the speed performance.

Even though some of the nonstandard security assumptions (e.g., SSSP⁵ [Lee 2011; Halevi and Ratha 2011]) in the Gentry's original scheme are later removed, there are still some open security issues about the FHE schemes. First one is the circular security of FHE. Circular security (aka KDM security), as mentioned earlier, keeps its own secret key secure by encrypting it with the public key. All known FHE schemes use Gentry's blueprint bootstrapping technique to obtain an unlimited FHE scheme. We know that the encryption of the secret key is also sent to the cloud to bootstrap the noisy ciphertexts and an eavesdropper can capture the encryption of secret key. Even though some SWHE and leveled-FHE schemes are proven as semantically secure, an unbounded FHE still has not been proven as semantically secure with respect to any function, so it does not guarantee that an adversary can not reveal the secret key from its encryption under public key. This unfortunate situation is still open to be proven. Moreover, although some SWHE schemes [Loftus et al. 2011] are proven as indistinguishable under non-adaptive chosen ciphertext attack (IND-CCA1), none of the unbounded FHE schemes is IND-CCA1 secure for now. (IND-CCA2 (adaptive) is

⁵Indeed, Moon Sung Lee showed that it is quite probable that SSSP challenges can be solved within two days [Lee 2011; Halevi and Ratha 2011].

not applicable to FHE because FHE itself requires to be malleable.) In brief, FHE still needs to be studied extensively to prove that it is secure enough.

FHE allows unlimited number of functions on encrypted data. However, limitations on the efficiency of the FHE schemes prompts researchers to find the SWHE schemes that can be good enough to use in real-life applications. Recently, homomorphic evaluation of one AES, which is a highly complex and nontrivial function, is reduced to 2 seconds [Gentry et al. 2012] and researchers are now focusing to improve this instead of trying to implement an FHE scheme, which is extremely slow for now.

The main process that increases the computational cost in FHE is the bootstrapping process. An unbounded FHE scheme that allows unlimited operations without bootstrapping is still an open problem. Indeed, the bootstrapping is necessary to decrease the noise in the evaluated ciphertexts. Hence, though a framework was suggested in [Nuida 2014], the design of noise-free FHE scheme is also one of the open problems. A noise-free FHE [Liu 2015] and an FHE without bootstrapping [Yagisawa 2015] are reported as insecure in [Wang].

Showing the existence of FHE instilled hope to solve other long waiting problems (applications) such as Functional Encryption (FE) (i.e., Identity-based encryption (IBE) and Attribute-based encryption (ABE)). Functional encryption basically controls the access over data while allowing computation on it according the features of identity or attribute. The aim of designing ABE or IBE based on FHE is to take the advantage of the functionality of two worlds. However, for now, there exists a few [Gentry et al. 2013; Clear and McGoldrick 2014; Clear and McGoldrick 2016; Wang et al. 2015a]. Another fruitful application of FHE is multi-party computation (MPC) which allows the computation of the function with multiple inputs from different users while keeping the inputs hidden. Even though there exist a few FHE-based MPC protocols [Damgård et al. 2012; López-Alt et al. 2012; Choudhury et al. 2013; Damgård et al. 2016] proposing these powerful and useful tools, unfortunately, their performances are not yet comparable with the conventional MPC approaches [Mood et al. 2014; Carter et al. 2013; Carter et al. 2015] because of the computational cost of the existing FHE schemes. However, FHE does not require any interaction, which reduces the complexity of the communication protocol significantly. However, there are still some gaps on how to realize those protocols. Furthermore, FHE itself can not perform homomorphic evaluation on independently encrypted data, i.e., multi-key FHE. some primitive result to deal with this issue was presented in [López-Alt et al. 2012]. However, the proposed scheme can only handle bounded number of user. When the cloud and number of connected devices are considered, the restriction may not be feasible. Hence, a multi-key FHE with unlimited number of users is another promising direction for future applications.

6. CONCLUSION

In today's always-on, Internet-centric world, privacy of data plays a more significant role than ever before. For highly sensitive systems such as online retail and e-banking, it is crucial to protect users' accounts and assets from malicious third parties. Nonetheless, today's norm is to encrypt the data and share the keys with the service provider, cloud operator, etc. In this model, the control over the privacy of the sensitive data is lost. The users or service providers with the key have exclusive rights on the data. Untrusted providers, cloud operators can keep sensitive data and its identifying credentials of users long after the user ends the relationship with the services. One promising direction to preserve the privacy of the data is to utilize homomorphic encryption (HE) schemes. HE is a special kind of encryption scheme, which allows any third party to operate on the encrypted data without decrypting it in advance. Indeed, the idea of HE has been around for over 30 years; however, the first plausible and achievable *Fully Ho-*

homomorphic Encryption (FHE) scheme was introduced by Craig Gentry in 2009. Since then, different FHE schemes demonstrated that FHE still needs to be improved significantly to be practical on every platform as they are very expensive for real-life applications. Hence, in this paper, we surveyed the HE and FHE schemes. Specifically, starting from the basics of HE, the details of the well-known *Partially HE* (PHE) and *Somewhat HE* (SWHE), which are important pillars of achieving FHE, were presented. Then, after classifying FHE schemes in the literature under four different categories, we presented the major FHE schemes with this classification. Moreover, we articulated the implementations and the new improvements in Gentry-type FHE schemes. Finally, we discussed promising research directions as well as lessons learned for interested researchers.

References

- Nitesh Aggarwal, Cp Gupta, and Iti Sharma. 2014. Fully Homomorphic symmetric scheme without bootstrapping. In *Cloud Computing and Internet of Things (CCIOT), 2014 International Conference on*. IEEE, 14–17.
- Carlos Aguilar-Melchor, Simon Fau, Caroline Fontaine, Guy Gogniat, and Renaud Sirdey. 2013. Recent advances in homomorphic encryption: A possible future for signal processing in the encrypted domain. *Signal Processing Magazine, IEEE* 30, 2 (2013), 108–117.
- S Sobitha Ahila and KL Shunmuganathan. 2014. State Of Art in Homomorphic Encryption Schemes. (2014).
- Miklós Ajtai. 1996. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. ACM, 99–108.
- Mufutau Akinwande. 2009. Advances in Homomorphic Cryptosystems. *J. UCS* 15, 3 (2009), 506–522.
- Martin R Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. 2015. Ciphers for MPC and FHE. In *Advances in Cryptology–EUROCRYPT 2015*. Springer, 430–454.
- Jacob Alperin-Sheriff and Chris Peikert. 2013. Practical bootstrapping in quasilinear time. In *Advances in Cryptology–CRYPTO 2013*. Springer, 1–20.
- Jacob Alperin-Sheriff and Chris Peikert. 2014. Faster bootstrapping with polynomial error. In *Advances in Cryptology–CRYPTO 2014*. Springer, 297–314.
- Josh Benaloh. 1994. Dense probabilistic encryption. In *Proceedings of the workshop on selected areas of cryptography*. 120–128.
- Josh Daniel Cohen Benaloh. 1987. *Verifiable secret-ballot elections*. Yale University. Department of Computer Science.
- Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. 2005. Evaluating 2-DNF formulas on ciphertexts. In *Theory of cryptography*. Springer, 325–341.
- Joppe W Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. 2013. Improved security for a ring-based fully homomorphic encryption scheme. In *Cryptography and Coding*. Springer, 45–64.
- Zvika Brakerski. 2012. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Advances in Cryptology–CRYPTO 2012*. Springer, 868–886.
- Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2011. Fully Homomorphic Encryption without Bootstrapping. Cryptology ePrint Archive, Report 2011/277. (2011). <http://eprint.iacr.org/>.
- Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2012. (Leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ACM, 309–325.
- Zvika Brakerski and Vinod Vaikuntanathan. 2011. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Advances in Cryptology–CRYPTO 2011*. Springer, 505–524.
- Zvika Brakerski and Vinod Vaikuntanathan. 2014a. Efficient fully homomorphic encryption from (standard) LWE. *SIAM J. Comput.* 43, 2 (2014), 831–871.
- Zvika Brakerski and Vinod Vaikuntanathan. 2014b. Lattice-based FHE as secure as PKE. In *Proceedings of the 5th conference on Innovations in theoretical computer science*. ACM, 1–12.
- Xiaolin Cao, Ciara Moore, Máire O'Neill, Elizabeth O'Sullivan, and Neil Hanley. 2013. Accelerating Fully Homomorphic Encryption over the Integers with Super-size Hardware Multiplier and Modular Reduction. *IACR Cryptology ePrint Archive* 2013 (2013), 616.
- Xiaolin Cao, Ciara Moore, Máire O'Neill, Neil Hanley, and Elizabeth O'Sullivan. 2014. High-speed fully homomorphic encryption over the integers. In *Financial Cryptography and Data Security*. Springer, 169–180.

- Henry Carter, Benjamin Mood, Patrick Traynor, and Kevin Butler. 2013. Secure outsourced garbled circuit evaluation for mobile devices. *Journal of Computer Security Preprint* (2013), 1–44.
- Henry Carter, Benjamin Mood, Patrick Traynor, and Kevin Butler. 2015. Outsourcing secure two-party computation as a black box. In *Cryptology and Network Security*. Springer, 214–222.
- Donald Donglong Chen, Nele Mentens, Frederik Vercauteren, Sujoy Sinha Roy, Ray CC Cheung, Derek Pao, and Ingrid Verbauwhede. 2015. High-speed polynomial multiplication architecture for ring-LWE and SHE cryptosystems. *Circuits and Systems I: Regular Papers, IEEE Transactions on* 62, 1 (2015), 157–166.
- Liquan Chen, Hongmei Ben, and Jie Huang. 2014. An Encryption Depth Optimization Scheme for Fully Homomorphic Encryption. In *Identification, Information and Knowledge in the Internet of Things (IIKI), 2014 International Conference on*. IEEE, 137–141.
- Zhigang Chen, Jian Wang, ZengNian Zhang, and Song Xinxia. 2014. A fully homomorphic encryption scheme with better key size. *Communications, China* 11, 9 (2014), 82–92.
- Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrede Lepoint, Mehdi Tibouchi, and Aaram Yun. 2013. Batch fully homomorphic encryption over the integers. In *Advances in Cryptology—EUROCRYPT 2013*. Springer, 315–335.
- Jung Hee Cheon, Hyunsook Hong, Moon Sung Lee, and Hansol Ryu. 2016. The polynomial approximate common divisor problem and its application to the fully homomorphic encryption. *Information Sciences* 326 (2016), 41–58.
- Jung Hee Cheon, Woo-Hwan Kim, and Hyun Soo Nam. 2006. Known-plaintext cryptanalysis of the Domingo-Ferrer algebraic privacy homomorphism scheme. *Inform. Process. Lett.* 97, 3 (2006), 118–123.
- Su-Jeong Choi, Simon R Blackburn, and Peter R Wild. 2007. Cryptanalysis of a homomorphic public-key cryptosystem over a finite group. *Journal of Mathematical Cryptology* 1, 4 (2007), 351.
- Ashish Choudhury, Jake Loftus, Emmanuela Orsini, Arpita Patra, and Nigel P Smart. 2013. Between a Rock and a Hard Place: Interpolating between MPC and FHE. In *Advances in Cryptology—ASIACRYPT 2013*. Springer, 221–240.
- Michael Clear and Ciarán McGoldrick. *Multi-identity and multi-key leveled fhe from learning with errors*. Technical Report. Cryptology ePrint Archive, Report 2014/798, 2014. <http://eprint.iacr.org>.
- Michael Clear and Ciarán McGoldrick. 2014. Bootstrappable identity-based fully homomorphic encryption. In *Cryptology and Network Security*. Springer, 1–19.
- Michael Clear and Ciaran McGoldrick. 2016. Attribute-Based Fully Homomorphic Encryption with a Bounded Number of Inputs. (2016).
- Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. 2013. Batch Fully Homomorphic Encryption over the Integers. Cryptology ePrint Archive, Report 2013/036. (2013). <http://eprint.iacr.org/>.
- Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. 2014. Scale-invariant fully homomorphic encryption over the integers. In *Public-Key Cryptography—PKC 2014*. Springer, 311–328.
- Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. 2011. Fully homomorphic encryption over the integers with shorter public keys. In *Advances in Cryptology—CRYPTO 2011*. Springer, 487–504.
- Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. 2012. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *Advances in Cryptology—EUROCRYPT 2012*. Springer, 446–464.
- David Bruce Cousins, John Golusky, Kurt Rohloff, and Daniel Sumorok. 2014. An FPGA co-processor implementation of Homomorphic Encryption. In *High Performance Extreme Computing Conference (HPEC), 2014 IEEE*. IEEE, 1–6.
- David Bruce Cousins, Kathrin Rohloff, Chris Peikert, and Richard Schantz. 2012. An update on SIPHER (scalable implementation of primitives for homomorphic encryption)—FPGA implementation using Simulink. In *High Performance Extreme Computing (HPEC), 2012 IEEE Conference on*. IEEE, 1–5.
- Wei Dai, Yarkin Doroz, and Berk Sunar. 2014. Accelerating NTRU based homomorphic encryption using GPUs. In *High Performance Extreme Computing Conference (HPEC), 2014 IEEE*. IEEE, 1–6.
- Wei Dai, Yarkin Doröz, and Berk Sunar. 2015. Accelerating SWHE based PIRs using GPUs. (2015).
- Ivan Damgård and Mads Jurik. 2001. A generalisation, a simplification and some applications of paillier’s probabilistic public-key system. In *Public Key Cryptography*. Springer, 119–136.
- Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. 2012. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology—CRYPTO 2012*. Springer, 643–662.
- Ivan Damgård, Antigoni Polychroniadou, and Vanishree Rao. 2016. Adaptively secure multi-party computation from lwe (via equivocal fhe). In *Public-Key Cryptography—PKC 2016*. Springer, 208–233.

- Whitfield Diffie and Martin E Hellman. 1976. New directions in cryptography. *Information Theory, IEEE Transactions on* 22, 6 (1976), 644–654.
- Josep Domingo-Ferrer. 2002. A provably secure additive and multiplicative privacy homomorphism*. In *Information security*. Springer, 471–483.
- Yarkin Doröz, Yin Hu, and Berk Sunar. 2014. Homomorphic AES Evaluation using NTRU. *IACR Cryptology ePrint Archive* 2014 (2014), 39.
- Yarkin Doröz, Erdiç Öztürk, Erkey Savaş, and Berk Sunar. 2015. Accelerating LTV Based Homomorphic Encryption in Reconfigurable Hardware. In *Cryptographic Hardware and Embedded Systems–CHES 2015*. Springer, 185–204.
- Yarkin Doroz, Erdinc Ozturk, and Berk Sunar. 2013a. Accelerating Fully Homomorphic Encryption in Hardware. (2013).
- Yarkin Doroz, E Ozturk, and Berk Sunar. 2013b. Evaluating the hardware performance of a million-bit multiplier. In *Digital System Design (DSD), 2013 Euromicro Conference on*. IEEE, 955–962.
- Yarkin Doröz, Aria Shahverdi, Thomas Eisenbarth, and Berk Sunar. 2014. Toward practical homomorphic evaluation of block ciphers using prince. In *Financial Cryptography and Data Security*. Springer, 208–220.
- Léo Ducas and Daniele Micciancio. 2014. A Fully Homomorphic Encryption library. <https://github.com/lducas/FHEW>. (2014). Accessed at December, 2015.
- Léo Ducas and Daniele Micciancio. 2015. FHEW: Bootstrapping Homomorphic Encryption in less than a second. In *Advances in Cryptology–EUROCRYPT 2015*. Springer, 617–640.
- Taher ElGamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in cryptology*. Springer, 10–18.
- Junfeng Fan and Frederik Vercauteren. 2012. Somewhat Practical Fully Homomorphic Encryption. *IACR Cryptology ePrint Archive* 2012 (2012), 144.
- Caroline Fontaine and Fabien Galand. 2007. A survey of homomorphic encryption for nonspecialists. *EURASIP Journal on Information Security* 2007 (2007), 15.
- Matteo Frigo and Steven G. Johnson. 2005. The Design and Implementation of FFTW3. *Proc. IEEE* 93, 2 (2005), 216–231. Special issue on “Program Generation, Optimization, and Platform Adaptation”.
- Steven D Galbraith. 2002. Elliptic curve Paillier schemes. *Journal of Cryptology* 15, 2 (2002), 129–138.
- Steven D Galbraith, Shishay W Gebregiyorgis, and Sean Murphy. 2016. Algorithms for the Approximate Common Divisor Problem. (2016).
- Craig Gentry. 2009. *A fully homomorphic encryption scheme*. Ph.D. Dissertation. Stanford University.
- Craig Gentry. 2010. Toward basing fully homomorphic encryption on worst-case hardness. In *Advances in Cryptology–CRYPTO 2010*. Springer, 116–137.
- Craig Gentry. 2012. personal communication. (2012).
- Craig Gentry. 2014. Computing on the edge of chaos: Structure and randomness in encrypted computation. In *Electronic Colloquium on Computational Complexity (ECCC)*, Vol. 21. 106.
- Craig Gentry and Shai Halevi. 2011. Implementing Gentry’s fully-homomorphic encryption scheme. In *Advances in Cryptology–EUROCRYPT 2011*. Springer, 129–148.
- Craig Gentry, Shai Halevi, Chris Peikert, and Nigel P Smart. 2012. Ring switching in BGV-style homomorphic encryption. In *Security and Cryptography for Networks*. Springer, 19–37.
- Craig Gentry, Shai Halevi, and Nigel P Smart. 2012. Homomorphic evaluation of the AES circuit. In *Advances in Cryptology–CRYPTO 2012*. Springer, 850–867.
- Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. 2010. A simple BGN-type cryptosystem from LWE. In *Advances in Cryptology–EUROCRYPT 2010*. Springer, 506–522.
- Craig Gentry, Amit Sahai, and Brent Waters. 2013. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013*. Springer, 75–92.
- Kristian Gjøsteen. 2004. Subgroup membership problems and public key cryptosystems. (2004).
- Oded Goldreich, Shafi Goldwasser, and Shai Halevi. 1997. Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology–CRYPTO’97*. Springer, 112–131.
- Shafi Goldwasser and Silvio Micali. 1982. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*. ACM, 365–377.
- Dima Grigoriev and Ilia Ponomarenko. 2006. Homomorphic public-key cryptosystems and encrypting boolean circuits. *Applicable Algebra in Engineering, Communication and Computing* 17, 3-4 (2006), 239–255.

- Shai Halevi and Nalini K. Ratha. 2011. Public Challenges for Fully-Homomorphic Encryption. http://researcher.ibm.com/researcher/view_group.php?id=1548. (2011). Accessed at March, 2016.
- Shai Halevi and Victor Shoup. 2013a. Design and implementation of a homomorphic-encryption library. *IBM Research (Manuscript)* (2013).
- Shai Halevi and Victor Shoup. 2013b. An Implementation of homomorphic encryption. <https://github.com/shaih/HElib>. (2013). Accessed at December, 2015.
- Shai Halevi and Victor Shoup. 2014. Algorithms in helib. In *Advances in Cryptology—CRYPTO 2014*. Springer, 554–571.
- Shai Halevi and Victor Shoup. 2015. Bootstrapping for helib. In *Advances in Cryptology—EUROCRYPT 2015*. Springer, 641–670.
- Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. 1998. NTRU: A ring-based public key cryptosystem. In *Algorithmic number theory*. Springer, 267–288.
- Jeffrey Hoffstein, Jill Pipher, Joseph H Silverman, and Joseph H Silverman. 2008. *An introduction to mathematical cryptography*. Vol. 1. Springer.
- Darko Hrestak and Stjepan Picek. 2014. Homomorphic encryption in the cloud. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on*. IEEE, 1400–1404.
- Josep Domingo i Ferrer. 1996. A new privacy homomorphism and applications. *Inform. Process. Lett.* 60, 5 (1996), 277–282.
- Yuval Ishai and Anat Paskin. 2007. Evaluating branching programs on encrypted data. In *Theory of Cryptography*. Springer, 575–594.
- Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. 2007. Multi-bit cryptosystems based on lattice problems. In *Public Key Cryptography—PKC 2007*. Springer, 315–329.
- Jinsu Kim, Moon Sung Lee, Aaram Yun, and Jung Hee Cheon. 2013. CRT-based Fully Homomorphic Encryption over the Integers. *IACR Cryptology ePrint Archive 2013* (2013), 57.
- Eyal Kushilevitz and Rafail Ostrovsky. 1997. Replication is not needed: Single database, computationally-private information retrieval. In *focs*. IEEE, 364.
- Moon Sung Lee. 2011. On the sparse subset sum problem from Gentry-Halevi’s implementation of fully homomorphic encryption. *IACR Cryptology ePrint Archive 2011* (2011), 567.
- Moon Sung Lee, Yongje Lee, Jung Hee Cheon, and Yunheung Paek. 2015. Accelerating bootstrapping in FHEW using GPUs. In *Application-specific Systems, Architectures and Processors (ASAP), 2015 IEEE 26th International Conference on*. IEEE, 128–135.
- Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. 1982. Factoring polynomials with rational coefficients. *Math. Ann.* 261, 4 (1982), 515–534.
- Tancrède Lepoint and Michael Naehrig. 2014. A comparison of the homomorphic encryption schemes FV and YASHE. In *Progress in Cryptology—AFRICACRYPT 2014*. Springer, 318–335.
- Bingxin Liu and Huapeng Wu. 2015. Efficient architecture and implementation for NTRUEncrypt system. In *Circuits and Systems (MWSCAS), 2015 IEEE 58th International Midwest Symposium on*. IEEE, 1–4.
- Dongxi Liu. 2015. Practical Fully Homomorphic Encryption without Noise Reduction. *IACR Cryptology ePrint Archive 2015* (2015), 468.
- Jake Loftus, Alexander May, Nigel P Smart, and Frederik Vercauteren. 2011. On CCA-secure somewhat homomorphic encryption. In *Selected Areas in Cryptography*. Springer, 55–72.
- Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. 2012. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. ACM, 1219–1234.
- Fujitsu Laboratories Ltd. 2013. Fujitsu Develops World’s First Homomorphic Encryption Technology that Enables Statistical Calculations and Biometric Authentication. <http://www.fujitsu.com/global/about/resources/news/press-releases/2013/0828-01.html>. (2013). press release dated August 5, 2013.
- Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2013. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)* 60, 6 (2013), 43.
- DS Malik, John N Mordeson, and MK Sen. 2007. *MTH 581-582 Introduction to Abstract Algebra*.
- Robert McMillan. 2013. Apple Finally Reveals How Long Siri Keeps Your Data. (April 2013). <http://www.wired.com/2013/04/siri-two-years/>
- Carlos Aguilar Melchor, Guilhem Castagnos, and Philippe Gaborit. 2008. Lattice-based homomorphic encryption of vector spaces. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*. IEEE, 1858–1862.

- Carlos Aguilar Melchor, Philippe Gaborit, and Javier Herranz. 2010. Additively homomorphic encryption with d-operand multiplications. In *Advances in Cryptology—CRYPTO 2010*. Springer, 138–154.
- Silvia Mella and Ruggero Susella. 2013. On the homomorphic computation of symmetric cryptographic primitives. In *Cryptography and Coding*. Springer, 28–44.
- Michal Mikuš. 2012. Experiments with the Plaintext Space in Gentry's Somewhat Homomorphic Scheme. *Tatra Mountains Mathematical Publications* 53, 1 (2012), 147–154.
- Hermann Minkowski. 1968. *Geometrie der zahlen*. Vol. 40.
- Benjamin Mood, Debayan Gupta, Kevin Butler, and Joan Feigenbaum. 2014. Reuse it or lose it: More efficient secure computation through reuse of encrypted values. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 582–596.
- Ciara Moore, Neil Hanley, John McAllister, Máire O'Neill, Elizabeth O'Sullivan, and Xiaolin Cao. 2013. Targeting FPGA DSP slices for a large integer multiplier for integer based FHE. In *Financial Cryptography and Data Security*. Springer, 226–237.
- Carl Moore, Maire O'Neill, Neil Hanley, and Elizabeth O'Sullivan. 2014a. Accelerating integer-based fully homomorphic encryption using Comba multiplication. In *Signal Processing Systems (SiPS), 2014 IEEE Workshop on*. IEEE, 1–6.
- Carl Moore, Maire O'Neill, Elizabeth O'Sullivan, Yarkin Doroz, and Berk Sunar. 2014b. Practical homomorphic encryption: A survey. In *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*. IEEE, 2792–2795.
- Gary L Mullen and Peter Jau-Shyong Shiue. 1994. *Finite fields: theory, applications, and algorithms*. Vol. 168. American Mathematical Soc.
- David Naccache and Jacques Stern. 1998. A new public key cryptosystem based on higher residues. In *Proceedings of the 5th ACM conference on Computer and communications security*. ACM, 59–66.
- Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. 2011. Can homomorphic encryption be practical?. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. ACM, 113–124.
- Koji Nuida. 2014. A Simple Framework for Noise-Free Construction of Fully Homomorphic Encryption from a Special Class of Non-Commutative Groups. *IACR Cryptology ePrint Archive* 2014 (2014), 97.
- Koji Nuida and Kaoru Kurosawa. 2015. (Batch) Fully Homomorphic Encryption over Integers for Non-Binary Message Spaces. In *Advances in Cryptology—EUROCRYPT 2015*. Springer, 537–555.
- Naoki Ogura, Go Yamamoto, Tetsutaro Kobayashi, and Shigenori Uchiyama. 2010. An improvement of key generation algorithm for Gentry's homomorphic encryption scheme. In *Advances in Information and Computer Security*. Springer, 70–83.
- Tatsuaki Okamoto and Shigenori Uchiyama. 1998. A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology—EUROCRYPT'98*. Springer, 308–318.
- E Öztürk, Yarkin Doröz, Berk Sunar, and E Savaş. 2015. *Accelerating somewhat homomorphic evaluation using FPGAs*. Technical Report. Cryptology ePrint Archive, Report 2015/294.
- Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptology—EUROCRYPT'99*. Springer, 223–238.
- Payal V Parmar, Shraddha B Padhar, Shafika N Patel, Niyatee I Bhatt, and Rutvij H Jhaveri. 2014. Survey of Various Homomorphic Encryption algorithms and Schemes. *International Journal of Computer Applications* 91, 8 (2014).
- Chris Peikert. 2015. *A decade of lattice cryptography*. Technical Report. Cryptology ePrint Archive, Report 2015/939.
- Henning Perl, Michael Brenner, and Matthew Smith. 2011a. An Implementation of the Fully Homomorphic Smart-Vercauteren Cryptosystem. <https://github.com/hcrypt-project/libScarab>. (2011). Accessed at December, 2015.
- Henning Perl, Michael Brenner, and Matthew Smith. 2011b. Poster: an implementation of the fully homomorphic Smart-Vercauteren crypto-system. In *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 837–840.
- Pedro Silveira Pisa, Michel Abdalla, and Otto Carlos Duarte. 2012. Somewhat homomorphic encryption scheme for arithmetic operations on large integers. In *Global Information Infrastructure and Networking Symposium (GIIS), 2012*. IEEE, 1–8.
- Thomas Pöppelmann, Michael Naehrig, Andrew Putnam, and Adrian Macias. 2015. Accelerating Homomorphic Evaluation on Reconfigurable Hardware. In *Cryptographic Hardware and Embedded Systems—CHES 2015*. Springer, 143–163.
- Y Govinda Ramaiah and G Vijaya Kumari. 2012a. Efficient public key homomorphic encryption over integer plaintexts. In *Information Security and Intelligence Control (ISIC), 2012 International Conference on*. IEEE, 123–128.

- Y Govinda Ramaiah and G Vijaya Kumari. 2012b. Towards Practical Homomorphic Encryption with Efficient Public Key Generation. *International Journal on Network Security* 3, 4 (2012), 10.
- Oded Regev. 2006. Lattice-based cryptography. In *Advances in Cryptology-CRYPTO 2006*. Springer, 131–141.
- Oded Regev. 2009. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)* 56, 6 (2009), 34.
- Ronald L Rivest, Len Adleman, and Michael L Dertouzos. 1978a. On data banks and privacy homomorphisms. *Foundations of secure computation* 4, 11 (1978), 169–180.
- Ronald L Rivest, Adi Shamir, and Len Adleman. 1978b. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (1978), 120–126.
- Kurt Rohloff and David Bruce Cousins. 2014. A scalable implementation of fully homomorphic encryption built on NTRU. In *Financial Cryptography and Data Security*. Springer, 221–234.
- Ron Rothblum. 2011. Homomorphic encryption: From private-key to public-key. In *Theory of cryptography*. Springer, 219–234.
- Sujoy Sinha Roy, Kimmo Järvinen, Frederik Vercauteren, Vassil Dimitrov, and Ingrid Verbauwhede. 2015. Modular Hardware Architecture for Somewhat Homomorphic Function Evaluation. (2015).
- T. Sander, A. Young, and M. Yung. 1999. Non-interactive cryptocomputing for NC1. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*. 554–566. DOI: <http://dx.doi.org/10.1109/SFFCS.1999.814630>
- Peter Scholl and Nigel P Smart. 2011. Improved key generation for Gentry’s fully homomorphic encryption Scheme. In *Cryptography and Coding*. Springer, 10–22.
- Robert Scott, Henry George Liddell, Henry Stuart Jones, and Roderick McKenzie. 1940. a new edition by HS Jones. (1940).
- Jaydip Sen. 2013. Homomorphic Encryption: Theory & Applications. *arXiv preprint arXiv:1305.5886* (2013).
- Alice Silverberg. 2013. Fully homomorphic encryption for mathematicians. *Women in Numbers 2: Research Directions in Number Theory* 606 (2013), 111.
- N.P. Smart and F. Vercauteren. 2011. Fully Homomorphic SIMD Operations. Cryptology ePrint Archive, Report 2011/133. (2011). <http://eprint.iacr.org/>.
- Nigel P Smart and Frederik Vercauteren. 2010. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography-PKC 2010*. Springer, 420–443.
- Nigel P Smart and Frederik Vercauteren. 2014. Fully homomorphic SIMD operations. *Designs, codes and cryptography* 71, 1 (2014), 57–81.
- Damien Stehlé and Ron Steinfeld. 2010. Faster fully homomorphic encryption. In *Advances in Cryptology-ASIACRYPT 2010*. Springer, 377–394.
- Damien Stehlé and Ron Steinfeld. 2011. Making NTRU as secure as worst-case problems over ideal lattices. In *Advances in Cryptology-EUROCRYPT 2011*. Springer, 27–47.
- Rainer Steinwandt and Willi Geiselmann. 2002. Cryptanalysis of Polly cracker. *Information Theory, IEEE Transactions on* 48, 11 (2002), 2990–2991.
- Zhou Tanping, Yang Xiaoyuan, Zhang Wei, and Wu Liqiang. 2015. Efficient Fully Homomorphic Encryption with Circularly Secure Key Switching Process. *a a 1* (2015), 1.
- Vinod Vaikuntanathan. 2011. Computing blindfolded: New developments in fully homomorphic encryption. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*. IEEE, 5–16.
- Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. 2010. Fully homomorphic encryption over the integers. In *Advances in cryptology-EUROCRYPT 2010*. Springer, 24–43.
- David Wagner. 2003. Cryptanalysis of an algebraic privacy homomorphism. In *Information Security*. Springer, 234–239.
- Fuqun Wang, Kunpeng Wang, and Bao Li. 2015a. An Efficient Leveled Identity-Based FHE. In *Network and System Security*. Springer, 303–315.
- Fuqun Wang, Kunpeng Wang, and Bao Li. 2015b. LWE-based FHE with better parameters. In *Advances in Information and Computer Security*. Springer, 175–192.
- Wei Wang, Zhilu Chen, and Xinming Huang. 2014. Accelerating leveled fully homomorphic encryption using GPU. In *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*. IEEE, 2800–2803.
- Wei Wang, Yin Hu, Lianmu Chen, Xinming Huang, and Berk Sunar. 2015. Exploring the feasibility of fully homomorphic encryption. *Computers, IEEE Transactions on* 64, 3 (2015), 698–706.
- Wei Wang and Xinming Huang. 2013. FPGA implementation of a large-number multiplier for fully homomorphic encryption. In *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*. IEEE, 2589–2592.

- Wei Wang, Xinming Huang, Niall Emmart, and Charles Weems. 2014. VLSI design of a large-number multiplier for fully homomorphic encryption. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 22, 9 (2014), 1879–1887.
- Yongge Wang. *Notes on Two Fully Homomorphic Encryption Schemes Without Bootstrapping*. Technical Report. Cryptology ePrint Archive, Report 2015/519, 2015. <http://eprint.iacr.org>.
- David J Wu. 2015. Fully homomorphic encryption: Cryptography’s holy grail. *XRDS: Crossroads, The ACM Magazine for Students* 21, 3 (2015), 24–29.
- Ting Wu, Hui Wang, and You-Ping Liu. 2012. Optimizations of Brakerski’s fully homomorphic encryption scheme. In *Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on*. IEEE, 2000–2005.
- Masahiro Yagisawa. 2015. Fully Homomorphic Encryption without bootstrapping. *IACR Cryptology ePrint Archive* 2015 (2015), 474.
- Hao-Miao Yang, Qi Xia, Xiao-fen Wang, and Dian-hua Tang. 2012. A new somewhat homomorphic encryption scheme over integers. In *Computer Distributed Control and Intelligent Environmental Monitoring (CDCIEM), 2012 International Conference on*. IEEE, 61–64.
- Andrew Chi-Chih Yao. 1982. Protocols for secure computations. In *FOCS*, Vol. 82. 160–164.
- Xiaojun Zhang, Chunxiang Xu, Chunhua Jin, Run Xie, and Jining Zhao. 2014. Efficient fully homomorphic encryption from RLWE with an extension to a threshold encryption scheme. *Future Generation Computer Systems* 36 (2014), 180–186.
- Zhenfei Zhang. 2014. Revisiting fully homomorphic encryption schemes and their cryptographic primitives. (2014).